OPTIMASI PENJADWALAN *SHIFT* JAGA DOKTER DI IGD MENGGUNAKAN ALGORITME GENETIKA (STUDI KASUS RUMAH SAKIT DI MALANG)

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Annisaa Amalia Safitri
NIM: 145150207111029

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2018

PENGESAHAN

OPTIMASI PENJADWALAN *SHIFT* JAGA DOKTER DI IGD MENGGUNAKAN ALGORITME GENETIKA
(STUDI KASUS RUMAH SAKIT DI MALANG)

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Annisaa Amalia Safitri
NIM: 145150207111029

Skripsi ini telah diuji dan dinyatakan lulus pada
26 Juli 2018
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Imam Cholissodin, S.Si, M.Kom	Lailil Muflikhah, S.Kom, M.Sc
NIK. 201201 850719 1 001	NIP. 19741113 200501 2 001

Mengetahui
Ketua Jurusan Teknik Informatika

Tri Astoto Kurniawan, S.T., M.T., Ph.D
NIP. 19710518 200312 1 001

PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar pustaka.

Apabila ternyata didalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 31 Juli 2018

Annisaa Amalia Safitri

NIM: 145150207111029

KATA PENGANTAR

Puji syukur kehadirat Tuhan Yang Maha Esa atas segala rahmat dan karunia yang telah diberikan-Nya sehingga penulis dapat menyelesaikan skripsi dengan judul "**Optimasi Penjadwalan *Shift* Jaga Dokter di IGD Menggunakan Algoritme Genetika (Studi Kasus Rumah Sakit di Malang)**" dapat diselesaikan tepat waktu. Tidak pernah lupa sholawat serta salam juga ditujukan kepada Rasulullah, Nabi Muhammad SAW dan para sahabat.

Skripsi ini memiliki tujuan sebagai tugas akhir penulis selama mengikuti masa perkuliahan dan juga sebagai salah satu syarat untuk mendapat gelar Sarjana Komputer dari Fakultas Ilmu Komputer, Universitas Brawijaya, Malang.

Penyelesain skripsi ini tentunya tak pernah terlepas dari bimbingan dan bantuan yang melibatkan banyak pihak. Terlepas dari hal tersebut penulis ingin mengungkapkan rasa terima kasih yang sebesar-besarnya kepada :

1. Bapak Wayan Firdaus Mahmudy, S.Si, M.T, Ph.D selaku Dekan di Fakultas Ilmu Komputer.
2. Bapak Tri Astoto Kurniawan, S.T, M.T, Ph.D selaku Ketua Jurusan Teknik Informatika di Fakultas Ilmu Komputer.
3. Bapak Agus Wahyu Widodo, S.T, M.Cs selaku ketua Program Studi Teknik Informatika di Fakultas Ilmu Komputer.
4. Bapak Imam Cholissodin, S.Si, M.Kom selaku dosen pembimbing pertama yang telah meluangkan waktunya untuk membimbing dan membantu penulis dalam penyusunan dan pengerjaan skripsi.
5. Ibu Lailil Muflikhah, S.Kom, M.Sc selaku dosen pembimbing kedua yang telah meluangkan waktunya untuk membimbing dan membantu penulis dalam penyusunan dan pengerjaan skripsi.
6. Bapak dan Ibu Dosen Fakultas Ilmu Komputer yang selama masa perkuliahan bersedia memberikan ilmunya beserta seluruh *civitas* kemahasiswaan dan akademik yang telah membantu untuk urusan akademik selama penulis menjadi mahasiswa.
7. Bapak Simson selaku pihak rumah sakit yang membantu proses pencarian data mengenai skripsi penulis.
8. Keluarga penulis terutama kedua orang tua, Ayah Sutrisno Hadi dan Mama Lailatus Sholicha yang selalu memberi doa, motivasi dan nasehat serta dukungan berupa moral dan materiil kepada penulis.
9. Eka, Ninda, Sarah, Adit, Elha, Danang, Isradi dan seluruh teman-teman selama 8 semester yang tidak bisa disebutkan penulis satu persatu yang telah membantu penulis beradaptasi dengan lingkungan kuliah saat masa perkuliahan.

10. Teman-teman SMP, SMA dan UAPSM UB yang mendukung serta memberi hiburan kepada penulis ketika penat atau sedang dalam masa sulit ketika mengerjakan skripsi.
11. Semua pihak yang membantu penulis dalam penyelesaian skipsi yang tidak bisa disebutkan satu persatu

Penulis menyadari bahwa penyusunan skripsi ini jauh dari kata sempurna yang tak lepas dari kekurangan dan kesalahan, sehingga saran dan kritik yang membangun sangat diharapkan oleh penulis. Akhir kata, semoga skripsi ini bermanfaat bagi semua pihak yang menggunakannya. Terima Kasih.

Malang, 31 Juli 2018

Penulis
nhiisaamalia@gmail.com

ABSTRAK

Annisaa Amalia Safitri, Optimasi Penjadwalan *Shift* Jaga Dokter Di IGD Menggunakan Algoritme Genetika (Studi Kasus Rumah Sakit Di Malang)

Pembimbing: imam Cholissodin, S.Si, M. Kom dan Lailil Muflikhah, S.Kom, M.Sc

Instalasi Gawat Darurat (IGD) merupakan unit terdepan yang bertugas menerima pasien saat terjadi keadaan darurat. Di dalam IGD, dokter harus tersedia selama 24 jam untuk menangani pasien yang datang kapanpun saat keadaan darurat sedang terjadi. Untuk menjaga kinerja dokter saat bekerja 24 jam didalam IGD, maka digunakan sistem jadwal jaga berupa sistem *shift*. Penjadwalan dilakukan selama 1 bulan dengan 11 orang dokter yang tiap harinya akan dibagi menjadi 3 *shift* kerja. Guna mengoptimasi dan membuat kombinasi terbaik dalam jadwal jaga dokter di IGD, maka dibuat sistem penjadwalan *shift* jaga dokter di IGD yang menggunakan algoritme genetika. Proses reproduksi yang dilakukan ada 2, yaitu proses *crossover* dengan menggunakan *extended intermediate crossover* dan proses mutasi dengan menggunakan *reciprocal exchange mutation*, serta digunakan proses seleksi yaitu proses *elitism*. Pengujian yang digunakan untuk penjadwalan *shift* jaga IGD ini ada 3 jenis pengujian. Pengujian pertama yaitu pengujian jumlah *popSize*, pengujian kedua yaitu pengujian nilai generasi, dan pengujian ketiga yaitu pengujian kombinasi *cr mr*. Dari hasil pengujian tersebut, dilakukan lagi pengujian untuk membandingkan nilai *fitness* dari sistem dengan nilai *fitness* dari data *real* yang diberikan oleh rumah sakit. Dan hasilnya menunjukkan bahwa nilai *fitness* dari sistem=11,111 lebih besar dari nilai *fitness* pada data real yang diberikan rumah sakit=7,692.

Kata kunci : optimasi, penjadwalan, *shift* jaga dokter, algoritme genetika

ABSTRACT

Annisaa Amalia Safitri, Optimization Of Doctor's Shifts Scheduling At ER Using Genetic Algorithm (Study Case Of Hospitals In Malang)

Supervisor: imam Cholissodin, S.Si, M. Kom dan Lailil Muflikhah, S.Kom, M.Sc

Emergency room (ER) is one of the units in a hospital who the first of receiving patients in case of an emergency. In ER, there are doctors who should be available for 24 hours to deal with patient who come everytime when an emergency form happen. To keep the performance of doctors who working 24 hours in ER, then we make a schedule that use with shift system. For 1 month scheduling, 11 doctors will split into 3 shift work in a day. In order to optimize and make the best combination in doctor's schedule at ER, then made the doctor's scheduling system in a ER using genetic algorithm. Reproductive process using 2 ways, first the process of crossover by using extended intermediate crossover and second the mutation process by using a reciprocal exchange mutation, and then will use the last process of algorithm and the name is elitism selection process. Testing that is used for doctor's scheduling system in a ER is there are 3 types of testing. The first test is testing the number of popSize, the second test is testing the value of generation, and the last test is combination of cr mr. From those results, do more testing to compare the fitness value of fitness values of the system with real data provided by the hospital. And the results show that the value of fitness of the system = 11,111 is greater than the value of the data on real fitness given hospital = 7,692.

***Keyword** : optimization, scheduling, doctor's schedule, genetic algorithm*

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR LAMPIRAN

BAB 1 PENDAHULUAN

1.1 Latar belakang

Instalasi Gawat Darurat (IGD) adalah *unit* dalam sebuah rumah sakit yang terdepan dalam menerima pasien saat pasien mengalami sebuah keadaan darurat (Jamil, 2015). Di dalam Instalasi Gawat Darurat (IGD), kondisi pasien dalam keadaan apapun baik yang parah maupun tidak harus tetap ditangani oleh dokter. Peran dokter, perawat, serta peralatan medis yang ada diperlukan dalam waktu 24 jam dan 7 hari untuk menangani pasien yang datang ke IGD agar mendapatkan pertolongan yang optimal (Ulya, 2017). Panjangnya jam kerja untuk tenaga medis (terutama bagi dokter), akan membutuhkan tenaga yang cukup besar. Di dalam kondisi ini, memungkinkan akibat pada kemudian hari akan timbul kondisi buruk pada dokter karena tenaganya terlalu di forsir untuk bekerja dalam 24 jam selama 7 hari penuh untuk menangani pasien di IGD (Ulya, 2017). Selain itu apabila kinerja dokter terlalu diforsir, maka akan menyebabkan kurang maksimalnya pelayanan terhadap para pasien di IGD. Berdasarkan permasalahan yang ada maka diperlukan sebuah penyelesaian.

Untuk mendapatkan solusi dari masalah-masalah yang telah disebutkan sebelumnya, maka diperlukan sebuah penjadwalan. Penjadwalan yang optimal adalah penjadwalan yang menerapkan keadilan untuk setiap dokter yang mendapatkan jadwal. Penyusunan jadwal *shift* jaga kurang optimal apabila dikerjakan secara manual. Karena mungkin saja meskipun di dalam IGD sudah terdapat jadwal *shift* jaga dokter, namun kadang nama dokter tersebut terdapat pada *shift* yang berurutan, ada 2 nama yang sama pada 1 shift, atau bahkan dalam 1 *shift* tidak ada dokter yang berjaga sama sekali (Rezaeiahari, 2017). Karena terbatasnya jumlah dokter yang ada di dalam rumah sakit yang dibandingkan dengan jumlah pasien serta jumlah shift yang ada (Ulya, 2017), dalam penyusunan jadwal *shift* jaga dokter ini, harus diperhatikan agar jadwal jaga mendapatkan hasil optimal, menguntungkan semua pihak, dan menghasilkan pelayanan yang terbaik untuk pasiennya (Atmasari, 2010).

Pada penerapan optimasi penjadwalan *shift* jaga dokter ini, akan digunakan algoritme genetika. Algoritme genetika adalah pendekatan secara komputasional untuk menyelesaikan masalah-masalah yang dibuat dengan proses biologi dari proses evolusi. Diharapkan dengan menggunakan algoritme genetika akan memperoleh optimasi penjadwalan yaitu kondisi dimana terbentuk kombinasi terbaik dokter yang berjaga pada IGD, serta tidak ada permasalahan bentrokan jadwal jaga pada 1 waktu (Sufarnap & Sudarto, 2011). Alasan lain penggunaan algoritme genetika adalah karena menurut (Marbun, 2013), nilai *fitness* serta waktu komputasi yang dilakukan dengan menggunakan algoritme genetika lebih baik dibandingkan dengan menggunakan metode *Particle Swarm Optimization (PSO)*. Penelitian ini didasarkan pada studi pustaka penelitian yang berhasil dilakukan oleh (Ilmi, 2014) yaitu dilakukannya optimasi penjadwalan perawat menggunakan algoritme genetika. Di dalam penelitian yang telah

dilakukan tersebut, algoritme genetika dapat membuat jadwal jaga yang dilakukan perawat menjadi lebih optimal. Masukkan yang diberikan dalam penelitian tersebut berupa *popsize*, generasi, *cr* dan *mr*. Untuk mendapatkan kualitas yang baik dari solusi penjadwalan yang dapat dihasilkan , maka dapat diukur dari nilai *fitness*-nya. Perhitungan *fitness* didapat dari jumlah penalti yang telah dikalikan dengan konstanta dan selanjutnya dihitung menggunakan rumus *fitness*. Semakin besar nilai *fitness* yang dapat dihasilkan dari proses perhitungan *constraint,* maka solusi yang dihasilkan akan semakin baik sedangkan apabila semakin kecil nilai *fitness* yang dapat dihasilkan dari proses perhitungan *constraint,* maka semakin buruk solusi yang akan dihasilkan. Dengan latar belakang masalah tersebut, maka kami membuat penelitian dengan judul "Optimasi Penjadwalan *Shift* Jaga Dokter di IGD Menggunakan Algoritme Genetika (Studi Kasus Rumah Sakit di Malang)". Harapan kami, dengan adanya sistem penjadwalan ini maka akan meningkatkan kinerja dokter dan memberikan hasil yang optimal terhadap pasien.

1.2 Rumusan masalah

Berdasarkan dari rumusan masalah yang telah dipaparkan, maka permasalahan yang dapat diangkat dari penelitian ini adalah:

1. Bagaimana parameter yang optimal dalam penerapan algoritme genetika pada masalah optimasi penjadwalan *shift* jaga dokter di IGD ?
2. Bagaimana hasil penerapan algoritme genetika pada optimasi penjadwalan *shift* jaga dokter di IGD ?

1.3 Tujuan

Tujuan penelitian ini adalah:

1. Mengetahui parameter yang optimal dalam penerapan algoritme genetika pada masalah optimasi penjadwalan *shift* jaga dokter di IGD.
2. Mengetahui hasil dari penerapan algoritme genetika pada optimasi penjadwalan *shift* jaga dokter di IGD.

1.4 Manfaat

Manfaat penelitian yang diharapkan dari penelitian ini adalah:

1. Sistem yang menggunakan Algoritme Genetika dapat menjadikan sebuah solusi dalam permasalahan penjadwalan s*hift* jaga dokter di IGD.
2. Hasil dalam sistem dapat meningkatkan efisiensi dan efektivitas dalam penyusunan penjadwalan s*hift* jaga dokter di IGD.

1.5 Batasan masalah

Batasan masalah dalam penelitian ini adalah:

1. Penjadwalan dibuat dalam periode 30 hari.
2. Jumlah *shift* kerja ada 3 *shift*. *Shift* pagi maksimal diisi oleh 4 orang, *shift* siang maksimal diisi oleh 3 orang, dan *shift* malam diisi oleh 1 orang.
3. Jumlah dokter berjumlah 11 orang.
4. Data yang digunakan merupakan data *real* dari salah satu IGD di rumah sakit di Malang.

1.6 Sistematika pembahasan

Laporan ini disusun dengan sistematika pembahasan sebagai berikut :

BAB 1 PENDAHULUAN

Membahas tentang latar belakang dilakukan sebuah penelitian sebagai alasan penulis melakukan penelitian, rumusan masalah yang selanjutnya akan dibahas, tujuan dan manfaat yang ingin dicapai, batasan masalah, sistematika dari Optimasi Penjadwalan *Shift* Jaga Dokter di IGD Menggunakan Algoritme Genetika.

BAB 2 LANDASAN KEPUSTAKAAN

Berisi pembahasan mengenai kajian pustaka atau referensi dan dasar teori yang berhubungan dengan Optimasi Penjadwalan *Shift* Jaga Dokter di IGD Menggunakan Algoritme Genetika

BAB 3 METODOLOGI

Berisi metode yang dilakukan dalam melakukan penelitian ini seperti studi literatur, analisis kebutuhan, metode pengambilan data, perancangan sistem, implementasi, pengujian dan pengambilan kesimpulan Optimasi Penjadwalan *Shift* Jaga Dokter di IGD Menggunakan Algoritme Genetika.

BAB 4 PERANCANGAN

Berisi metode yang akan dilakukan dalam melakukan penelitian, seperti analisa kebutuhan perangkat lunak, perancangan Optimasi Penjadwalan *Shift* Jaga Dokter di IGD Menggunakan Algoritme Genetika.

BAB 5 IMPLEMENTASI

Bab ini berisi proses implementasi yang dilakukan. Implementasi ini terdiri dari spesifikasi sistem, implementasi algoritme dan implementasi *interface* dari Optimasi Penjadwalan *Shift* Jaga Dokter di IGD Menggunakan Algoritme Genetika.

BAB 6 PENGUJIAN DAN ANALISIS

Memuat hasil pengujian dan analisis terhadap sistem yang telah direalisasikan dalam Optimasi Penjadwalan *Shift* Jaga Dokter di IGD Menggunakan Algoritme Genetika. Pada bab ini dijelaskan proses pengujian dilakukan dengan beberapa cara yaitu pertama menguji fungsionalitas sistem, kemudian pengujian dengan membandingkan hasil perhitungan manual dengan proses output dari sistem yang telah dibuat, dan pengujian dengan batas alternative.

BAB 7 PENUTUP

Bab ini berisi kesimpulan dan saran dari keseluruhan laporan diatas serta. Saran berisi tulisan untuk pengembangan selanjutnya dan kesimpulan berisi hasil kesimpulan yang didapatkan dari proses pengembangan aplikasi Optimasi Penjadwalan *Shift* Jaga Dokter di IGD Menggunakan Algoritme Genetika.

BAB 2 LANDASAN KEPUSTAKAAN

2.1 Kajian Pustaka

Unit Gawat Darurat (UGD) adalah unit dalam sebuah rumah sakit yang terdepan dalam menerima pasien saat pasien mengalami sebuah keadaan darurat. Di dalam Unit Gawat Darurat (UGD), peran dokter, perawat, serta peralatan medis yang ada diperlukan dalam waktu 24 jam dan 7 hari untuk menangani pasien yang datang ke UGD agar mendapatkan pertolongan yang optimal. Bagi perawat, jadwalnya jam kerja pada UGD dikhawatirkan akan memberikan dampak yang buruk terhadap kualitas kinerja, kondisi fisik serta kehidupan sosial. Untuk mengurangi resiko tersebut, salah satu cara yang dilakukan pihak rumah sakit adalah membuat penjadwalan jam kerja yang dapat membagi jam kerja perawat secara adil terhadap semua perawat yang ada. Untuk menyelesaikan permasalahan tersebut, penjadwalan perawat pada UGD diselesaikan dengan *Genetic Algorithm* (GA) dan dua tingkat pemodelan. Pemodelan tingkat pertama digunakan untuk penentuan hari kerja dan libur dari perawat, sedangkan pemodelan tingkat kedua digunakan untuk penentuan *shift* kerja pada perawat. Hasil yang di dapatkan dari penelitian tersebut berupa nilai *fitness* terbaik yang didapatkan pada uji coba ke 6 dari nilai maksimal generasi sebesar 100. Mutasi gen yang dilakukan sebanyak 2 gen serta jumlah titik potong yang dilakukan terhadap *crossover* sebanyak 5 titik. Dan bobot yang paling seimbang dari dilakukannya pembobotan nilai *fitness* adalah bobot 1 untuk nilai varian serta nilai 1.5 untuk nilai rasio (Ulya, 2017).

Penjadwalan yang dilakukan untuk menentukan jadwal jaga dokter residen merupakan hal yang tidak mudah. Tingkat kerumitan penyusunan jadwal pada jadwal jaga dokter residen memiliki kesamaan dengan penentuan jadwal kegiatan belajar mengajar pada sekolah atau universitas. Pada permasalahan ini, terkadang dokter residen mendapat jadwal yang sama pada rumah sakit yang berbeda. Selain dilihat dari sisi senior tidaknya seorang dokter, penjadwalan dapat dilihat dari sisi dokter residen yang masih belajar untuk menyelesaikan banyak kasus berbeda pada setiap pasien yang masuk ke dalam rumah sakit. Dokter residen sendiri dapat digolongkan menurut berapa *stase* yang sudah dijalani selama menjadi dokter residen serta dipertimbangkan juga tingkat berapa atau semester berapa dokter residen tersebut. Karena tiap *stase* yang dilalui dokter residen mempengaruhi sudah berapa kasus yang ditangani, sudah berapa pasien yang ditangani, dan hal tersebut akan menjadi dasar untuk membuat jadwal jaga dokter residen. Penjadwalan dilakukan agar dengan jumlah dokter residen yang ada, jadwal jaga yang dilakukan dapat optimal, tidak ada yang memiliki jadwal jaga yang sama pada rumah sakit yang berbeda. Pada permasalahan penjadwalan dokter residen ini digunakan pendekatan algoritme genetika yang dianggap paling optimal untuk menyelesaikan masalah. Hasil yang didapatkan dari penelitiaan ini berupa penjadwalan jaga dokter residen yang optimal baik dari segi posisi jaga, waktu jaga, ruangan tempat untuk berjaga, serta meminimalkan tingginya frekuensi jaga dari dokter residen (Sufarnap & Sudarto, 2011).

Optimasi penjadwalan telah dilakukan oleh (Ilmi, 2014). Penjadwalan perawat adalah masalah yang cukup rumit dan krusial dalam sebuah rumah sakit khususnya pada ruangan *Intensive Care Unit* (*ICU).* Di dalam ruangan *ICU*, pasien yang dirawat adalah pasien yang memiliki perawatan khusus dan harus diutamakan. Akibat dari banyaknya jam kerja yang dipakai, merupakan suatu masalah kritis pada suatu rumah sakit terutama pada ruang *ICU* dengan pasien yang membutuhkan perawatan khusus. Dampak dari panjangnya jam kerja perawat serta tugas yang banyak dikhawatirkan akan memberikan dampak yang buruk terhadap kualitas kinerja, kondisi fisik serta kehidupan sosial. Untuk mengurangi resiko tersebut, salah satu cara yang dilakukan pihak rumah sakit adalah membuat penjadwalan jam kerja yang dapat membagi jam kerja perawat secara adil terhadap semua perawat yang ada. Dalam penelitian ini, diterapkan algoritme genetika untuk menyelesaikan mengoptimalkan permasalahan penjadwalan perawat. Digunakan representasi permutasi bilangan integer dengan panjang kromosom 360 yang setiap angka pada gennya merepresentasikan nomor id perawat. Metode crossover yang digunakan yaitu *one cut-point crossover*, metode mutasi *reciprocal exchange mutation* dan diseleksi dengan *elitism selection*. Hasil yang didapatkan adalah penjadwalan selama 1 bulan dengan parameter yang optimal dengan 200 individu (rata-rata *fitness* 0.80094) serta 150 generasi (rata-rata *fitness* 2.13674). Kombinasi *cr* dan *mr* yang di dapatkan adalah masing-masing 0.5 (rata-rata *fitness* 3.4266).

Optimasi penjadwalan lainnya juga dilakukan oleh (Rezaeiahari & Khasawneh, 2017). Di dalam penelitiannya, dilakukan optimasi terhadap wisatawan yang melakukan perjalanan ke pusat medis untuk melakukan perawatan medis. Tujuan dilakukannya optimasi yaitu untuk (1) meminimalkan penyimpangan atau kesalahan waktu saat pasien akan memulai perjalanan, dan (2) meminimalkan waktu tunggu antar pasien di dalam pusat medis tujuan (*Destination Medical Center*). Permasalahan penjadwalan disini, digunakan perbandingan hasil metode antara *MPSO* (Algoritme penambangan yang diterapkan dalam *PSO*), PSO^{LS} (*PSO* yang menggunakan *local search* untuk mencari *global best*), dan *SA* (Teknik pencarian yang menggunakan pencarian lokal untuk mendapatkan hasil dengan cara membandingkan dengan solusi sebelumnya). Parameter dari algoritme ini mengunakan desain *Taguchi* yang berfungsi untuk memperbaiki kualitas penjadwalan. Pada penerapan metode *hybrid* ini yang memiliki hasil presentasi selisih relative (*RPD*) terbaik didapatkan oleh *MPSO* yaitu 0.10 dan hasil presentasi selisih relative (*RPD*) terburuk didapatkan oleh SA yaitu 1.18.

Pada penelitian ini, akan dilakukan optimasi terhadap penjadwalan *shift* jaga dokter di IGD. Optimasi dilakukan agar jadwal yang dibuat lebih optimal serta memberikan keuntungan bagi banyak pihak terutama dokter dan pasien. Algoritme yang digunakan untuk menyelesaikan permasalahan optimasi terhadap penjadwalan *shift* jaga dokter di IGD adalah algoritme genetika. Proses *crossover* nya menggunakan *extended intermediate crossover*, mutasinya menggunakan *reciprocal exchange mutation*, dan seleksinya menggunakan seleksi *elitism*.

Tabel 2.1 Kajian Pustaka

JUDUL	Objek Masukkan dan Parameter	Metode Proses	Keluaran Hasil Penelitian
Penggunaan Algoritme Genetik Dengan Pemodelan Dua Tingkat Dalam Permasalahan Penjadwalan Perawat Pada Unit Gawat Darurat Rumah Sakit Umum Xyz Surabaya	Jumlah perawat serta jumlah hari libur.	Algoritme Genetika - Kromosom merepresentasikan jadwal kerja N perawat selama T hari - Mutasi gen sebanyak 2 gen serta jumlah titik potong sebanyak 5 titik. Dan bobot yang paling seimbang dari nilai *fitness* adalah bobot 1 untuk nilai varian serta nilai 1.5 untuk nilai rasio - Hasil terbaik adalah *nilai* yang dilakukan pada uji coba ke 6.	Jadwal jaga perawat dengan porsi jam kerja dan hari libur yang seimbang sesuai dengan yang diinginkan oleh perawat.
Analisis Optimasi Penjadwalan Jaga Dokter Residen Penyakit Dalam Pada Rumah Sakit Pendidikan	Data yang perlu disertakan untuk setiap Levelisasi yang ada adalah semester residen, jumlah *stase* yang sudah dilalui, dokter residen serta ruangan.	Algoritme genetika - Berdasarkan urutan dari Tabel Prioritas Levelisasi Jaga, setiap level jaga akan dijadwalkan ke dalam Tabel Jadwal Level Jaga secara acak - Rumus *fitness* yang digunakan adalah $Fitness = \frac{1}{B1xF1+B2xF2+B3xF3+B4xF4+B5xF5+B6xF6}$	Tabel Jaga berisikan daftar jaga seluruh dokter residen akan dilaksanakan pada bulan berikutnya pada rumah sakit pendidikan.
Optimasi Penjadwalan Perawat Menggunakan Algoritme Genetika	Masukkan parameter algoritme genetika berupa *popsize*, generasi, *cr* dan *mr*.	Algoritme Genetika Representasi kromosomnya yaitu : setiap 4 kolom menunjukkan *shift* dan sehingga 12 kolom menunjukkan kebutuhan perawat dalam satu hari seterusnya sampai hari	Hasil akhir berupa jadwal jaga perawat pada ruang ICU selama 1 bulan

		ke 30 dan berjumlah 360.	
An Optimization Model for Scheduling Patients in Destination Medical Center	Parameter dari algoritme ini mengunakan desain *Taguchi* dengan ukuran masalah yang berbeda.	*MPSO, PSO^{LS},* dan *SA*. Rumus *Relative Precentage Deviation* (*RPD*) : $RPD = \frac{Alg.sol - Best.sol}{Best.sol} x100$	Hasil presentasi selisih relative (*RPD*) terbaik didapatkan oleh *MPSO* yaitu 0.10.
Optimasi Penjadwalan *Shift* Jaga Dokter di IGD Menggunakan Algoritme Genetika (Studi Kasus di Rumah Sakit Malang)	Masukkan parameter algoritme genetika berupa *popsize*, generasi, *cr* dan *mr*.	Algoritme Genetika Representasi kromosomnya yaitu : terdapat 4 kolom untuk *shift* pagi, 3 kolom untuk *shift* siang, dan 1 kolom untuk *shift* malam yang dibuat selama 30 hari. Jadi total jumlah gen dalam 1 kromosom adalah 240.	Hasil akhir berupa jadwal *shift* jaga dokter di IGD

2.2 Penjadwalan

Penjadwalan di dalam KBBI adalah proses untuk mengatur suatu urutan dalam kegiatan dengan pemilihan waktu yang tepat. Penjadwalan dapat diartikan juga sebagai suatu kegiatan alokasi sumber daya dengan memiliki kendala (batasan) yang diberikan kepada suatu objek seperti di ruang-waktu, sedemikian rupa untuk memenuhi sedekat mungkin set tujuan yang diinginkan (Suhartono, 2015). Penjadwalan dapat meliputi alokasi dari suatu sarana kegiatan dan mengenai hal-hal yang akan dilakukan dalam suatu kegiatan (Rifai, 2011).

Penjadwalan da1lam pekerjaan merupakan pengalokasian sumber daya manusia pada suatu tempat kerja tertentu dengan waktu dan tempat yang telah ditentukan dalam melaksanakan pekerjaan-pekerjaan yang telah direncanakan untuk mencapai tujuan yang diinginkan oleh perusahaan (Dhuha & Suseno, 2017). *Shift* kerja didefinisikan sebagai periode waktu 24 jam yang satu atau kelompok orang dijadwalkan atau diatur untuk bekerja di tempat kerja, *Oxford Advanced Learner's Dictionary* tahun 2005 me mendefinisikan *shift* kerja sebagai suatu periode waktu yang dikerjakan oleh sekompok pekerja yang mulai bekerja ketika kelompok sebelumnya telah selesai (Dhuha & Suseno, 2017).

Penjadwalan yang baik adalah penjadwalan yang telah memiliki aturan penjadwalan yang pasti dan mempunyai tujuan yang jelas. Sehingga penjadwalan tersebut dapat digunakan untuk menyelesaikan masalah yang sebelumnya belum terselesaikan dan dapat dilakukan perubahan terhadap penjadwalan yang dibuat.

2.3 Instalasi Gawat Darurat (IGD)

Instalasi Gawat Darurat (IGD) adalah unit dalam sebuah rumah sakit yang terdepan dalam menerima pasien saat pasien mengalami sebuah keadaan darurat (Jamil, 2015). Di dalam Instalasi Gawat Darurat (IGD), kondisi pasien dalam keadaan apapun baik yang parah maupun tidak harus tetap ditangani oleh dokter. Di sini peran dokter, perawat, serta peralatan medis yang ada diperlukan dalam waktu 24 jam dan 7 hari untuk menangani pasien yang datang ke IGD agar mendapatkan pertolongan yang optimal (Ulya, 2017). IGD berperan sebagai gerbang utama untuk masuknya penderita gawat darurat. Kemampuan dari fasilitas kesehatan terutama rumah sakit yang secara keseluruhan sebagai pusat rujukan penderita dari rumah sakit lain dapat dicerminkan dari kemampuan instalasi gawat darurat ini (Malliarou, 2014).

2.4 Dokter

Dokter adalah seseorang yang telah lulus dari pendidikan kedokteran baik dari lulusan pendidikan dalam negeri maupun lulusan dari pendidikan di luar negeri yang telah diakui profesinya oleh Pemerintah Republik Indonesia dan sesuai dengan peraturan perundang-undangan yang berlaku. Dokter berperan menjadi salah satu komponen yang terpenting dalam suatu instansi atau pusat kesehatan masyarakat yang akan terkait secara langsung dengan pasien untuk melakukan pelayanan kesehatan serta menjamin mutu dari pelayanan kesehatan yang diberikan. Ilmu pengetahuan, keterampilan, serta sikap dan perilaku merupakan sebuah kompetensi yang harus didapatkan selama pendidikan akan menjadi landasan utama bagi dokter untuk dapat melakukan tindakan pada dunia kedokteran dalam upaya melakukan (Mutakhiroh, et al., 2007)pelayanan kesehatan. Pendidikan kedokteran pada dasarnya bertujuan untuk meningkatkan mutu kesehatan bagi seluruh masyarakat (Indonesia, 2006).

2.5 Algoritme Genetika

Algoritme genetika adalah algoritme pencarian berdasarkan mekanisme seleksi alami dan pewarisan genetik (Gen & Cheng, 1997). Algoritme genetika akan membuat kombinasi antara suatu deretan struktur dengan melakukan pertukaran informasi acak ke bentuk algoritme pencarian dengan beberapa perubahan bakat yang ada pada manusia. Dari generasi ke generasi, himpunan yang baru dari deretan individu dibuat atas dasar kecocokan pada generasi sebelumnya (Mutakhiroh, et al., 2007). Siklus pembuatan algoritme genetika untuk mencari solusi terbaik memiliki beberapa proses, seperti inisialisasi; reproduksi; evaluasi dan seleksi. Di dalam evaluasi terdapat proses perhitungan *fitness*. Untuk melakukan evaluasi nilai pada *fitness* dari kromosom, dapat dilakukan langkah-langkah berikut (Mahmudy & Rahman, 2011):

1) Ambil nilai asli dari tiap individu dalam populasi.
2) Membuat evaluasi dari nilai fungsi tujuan *f(x)*.
3) Membuat konversi dari nilai fungsi tujuan menjadi nilai *fitness*.

Semakin besar nilai dari *fitness* suatu individu, maka akan semakin baik juga solusi yang didapatkan dari individu tersebut. Nilai pada *fitness* ini akan digunakan untuk memilih *n* individu yang paling baik yang dapat bertahan hidup untuk generasi selanjutnya. (Mahmudy & Rahman, 2011). Ketika sudah diketahui nilai *fitness* dari semua individu, dilakukan seleksi untuk memilih individu terbaik. Seleksi merupakan suatu proses yang akan digunakan untuk memilih individu mana yang akan digunakan untuk proses pada generasi berikutnya dengan menggunakan pertimbangan nilai *fitness* (Yaqin & Lisbiantoro, 2012).

2.5.1 Proses Algoritme Genetika

2.5.1.1 Inisialisasi

Individu-individu diciptakan secara acak memiliki kromosom. Kromosom mewakili solusi permasalahan. Representasi kromosom yang digunakan adalah representasi permutasi. Pada inisialisasi ini, juga terdapat populasi yang merepresentasikan nomor *ID* dokter. Contoh kromosom yang menggunakan representasi permutasi terdapat pada Gambar 2.1.

5	2	3	4	6	7	1	8

Gambar 2.1 Contoh Kromosom yang menggunakan Representasi Permutasi

2.5.1.2 Reproduksi

Untuk menghasilkan *offspring* dari individu pada populasi. Reproduksi yang akan digunakan pada penelitian ini ada 2, yaitu *crossover* dan mutasi.

a. *Crossover*

Crossover yang digunakan dalam penelitian ini berupa *extended intermediate crossover.* Pemilihan *parent* yang digunakan dilakukan secara acak. Dimisalkan P1 dan P3 adalah *parent* yang telah dipilih secara acak. Kemudian dilakukan reproduksi *crossover* yang nantinya akan menghasilkan *offspring* berupa C1 dan C2. Nilai *offspring* dibangkitkan dengan menggunakan Persamaan 2.1 dan 2.2.

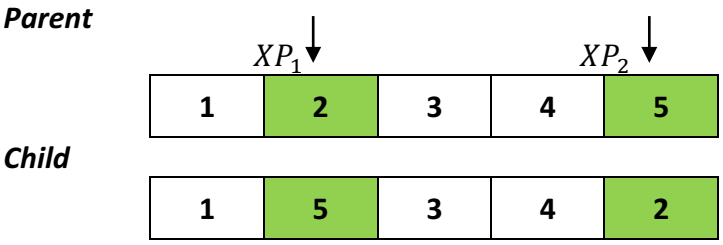
$$C_1 = P_1 + \alpha(P_2 - P_1) \quad (2.1)$$

$$C_2 = P_2 + \alpha(P_1 - P_2) \quad (2.2)$$

Dengan ketentuan nilai α yang dipilih secara acak pada interval [-0.25, 1.25].

b. *Mutation*

Mutasi yang digunakan adalah *reciprocal exchange mutation* yang ilustrasinya akan dijelaskan pada Gambar 2.2.



Gambar 2.2 Contoh Mutasi *Reciprocal Exchange*

2.5.1.3 Evaluasi

Untuk menghitung *fitness* setiap kromosom. Semakin besar fungsi *fitness* semakin baik kromosom untuk menjadi calon solusi. Contoh fungsi *fitness* yang digunakan oleh (Sufarnap & Sudarto, 2011) pada penelitian yang telah dilakukan sesuai dengan Persamaan 2.3.

$$Fitness = \frac{1}{B1xF1+B2xF2+B3xF3+B4xF4+B5xF5+B6xF6} \quad (2.3)$$

dimana:

*F*1 = Banyaknya Levelisasi Jaga yang dipecah
*F*2 = Banyaknya waktu jaga yang kosong
*F*3 = Banyaknya frekuensi jaga yang tinggi dari seorang dokter residen
*F*4 = Banyaknya frekuensi posisi jaga yang tinggi dari satu posisi jaga
*F*5 = Banyaknya level jaga yang berdekatan
*F*6 = Banyaknya level jaga yang berjauhan
*B*1 = Bobot pemecahan levelisasi jaga
*B*2 = Bobot waktu jaga yang kosong
*B*3 = Bobot frekuensi jaga dokter residen
*B*4 = Bobot frekuensi level posisi
*B*5 = Bobot level jaga yang berdekatan
*B*6 = Bobot level jaga yang berjauhan

2.5.1.4 Seleksi

Untuk memilih individu dari himpunan populasi dan *offspring* untuk bertahan pada generasi berikutnya. Dalam seleksi atau memilih individu untuk bertahan, digunakan fungsi probabilitas. Metode seleksi yang biasa digunakan yaitu *binary tournament*, *roulette wheel* dan *elitism*. Seleksi yang digunakan dalam penelitian ini adalah seleksi *elitism*. Seleksi tersebut merupakan seleksi dengan memilih hasil individu yang terbaik yang nantinya dapat masuk ke generasi selanjutnya. Di dalam Gambar 2.4 akan diilustrasikan seleksi elitism.

Individu dengan *popSize 5*

Individu	*Fitness*
P_1	10
P_2	8

P_3	4
P_4	7
P_5	6

Offspring (child)

Individu	*Fitness*
C_1	3
C_2	8
C_3	5

Individu yang lolos untuk generasi berikutnya

P(t+1)	Asal P(t)	*Fitness*
P_1	P_1	10
P_2	P_2	8
P_3	C_2	8
P_4	P_4	7
P_5	P_5	6

Gambar 2.3 Contoh Seleksi *Elitism*

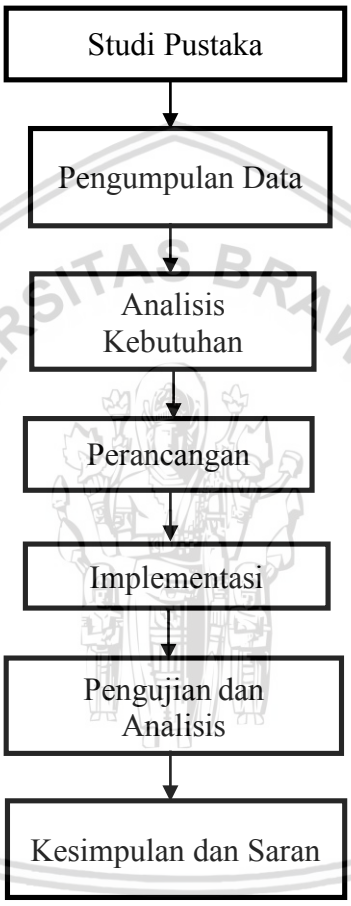
2.5.1.5 Melakukan Iterasi

Menurut (Syarif, 2014) iterasi dilakukan hinggan memenuhi *termination* (kondisi berhenti). Kriterianya adalah:

- Mencapai generasi *n*
  Nilai n telah ditentukan berdasarkan eksperimen yang dilakukan. Biasa nilai *n* akan semakin besar apabila semakin tinggi ukuran dan kompleksitas.
- Setelah beberapa generasi *n* tidak ditemukan solusi yang lebih baik (signifikan).
- Mencapai satuan waktu yang ditentukan
  Biasanya dilakukan ketika ingin membandingkan performa algoritme.
- Mendapatkan individu terbaik
  Setelah dilakukan beberapa iterasi ataupun setelah mendapatkan yang terbaik.

BAB 3 METODOLOGI

Agar hasil yang diinginkan tidak menyimpang dari tujuan, metodologi penelitian digunakan sebagai dasar pedoman dalam pelaksanaan penelitian. Metodologi penelitian yang digunakan dalam penyusunan proyek akhir ini akan melalui beberapa tahapan yang membentuk sebuah alur yang sistematis. Tahapan-tahapan yang akan dilalui akan digambarkan dengan *flowchart* pada Gambar 3.1.



Gambar 3.1 Flowchart Metodologi Penelitian Tugas Akhir

3.1 Studi Pustaka

Studi ini dilakukan berdasarkan proses membaca dan pencarian informasi dari literatur, jurnal, skripsi, dan hasil penelitian serupa mengenai pemahaman penjadwalan, algoritme genetika sebagai algoritme yang digunakan, pengertian IGD, serta pengertian dokter.

3.2 Pengumpulan Data

Pada saat proses pengerjaan skripsi ini, diperlukan proses pengumpulan data. Pengumpulan data ini didapatkan dari mahasiswa kedokteran yang sedang melakukan *stase* jaga di IGD. Data yang digunakan berisi nama, jumlah *shift* jaga, dan jam dari *shift* jaga. Jadwal jaga didapatkan dari salah satu Rumah Sakit Swasta di Malang. Jadwal yang diberikan adalah jadwal selama 1 bulan pada bulan Februari tahun 2018.

3.3 Analisis Kebutuhan

Analisis kebutuhan dilaksanakan untuk mengetahui secara keseluruhan kebutuhan yang harus ada dalam merancang, mengimplementasi dan menguji. Kebutuhan minimal yang digunakan dalam perancangan, implementasi dan pengujian adalah sebagai berikut:

1. *Hardware* yang digunakan meliputi:
   - Laptop dengan Prosesor 1,6 GHz *Intel Core* i5
   - *RAM* 8 *GB*
2. *Software* yang digunakan meliputi:
   - *macOS* sebagai sistem operasi
   - *MAMP* sebagai *server localhost*
   - *Sublime Text* sebagai *IDE*
   - *Microsoft Office* sebagai perhitungan manual dan penempatan data

3.4 Perancangan

Perancangan arsitektur perlu dilakukan untuk mempermudah pembutan perangkat lunak (*software*). Di dalam perancangan ini, akan ditunjukkan perhitungan manual menggunakan algoritme genetika sebagai metodenya, perancangan pengujian dan perancangan antarmuka akan akan dibuat gambarannya.

3.5 Implementasi

Implementasi Optimasi Penjadwalan *Shift* jaga dokter di IGD adalah menerapkan hal-hal yang telah didapat dalam proses studi literatur dan representasi perancangan. Fase-fase yang ada di dalam implementasi antara lain:

1. Pembuatan program dengan bahasa *php* menggunakan *IDE Sublime Text* dan *MAMP* sebagai *server*-nya.
2. Penerapan algoritme genetika dalam program yang dibuat.

3.6 Pengujian dan Analisis

Pengujian data berfungi untuk mengukur validasi hasil Optimasi Penjadwalan *Shift* jaga dokter di IGD. Jenis pengujian yang dilakukan adalah menggunakan pengujian hasil konvergensi banyaknya generasi, nilai *popSize*, dan pengujian terhadap nilai *cr* dan *mr* yang diinputkan pada awal program.

3.7 Kesimpulan dan Saran

Setelah melakukan semua tahapan penelitian, maka akan disimpulkan hasil dari penelitian yang diusulkan ini. Kesimpulan penelitian ini bisa berisi hasil penjadwalan *shift jaga*, hasil reprsentasi kromosom, hasil nilai *fitness,* dan perbandingan nilai aktual dengan nilai hasil program serta mengetahui kelebihan serta kekurangan dari penelitian ini.

Saran didasarkan atas hasil temuan pada penelitian yang dilakukan. Saran berisi kekurangan yang ada penelitian ini dan memberikan penjelasan apa yang dapat diperbaiki untuk penelitian selanjutnya.

BAB 4 PERANCANGAN

Pada bab ini akan dibahas mengenai formulasi permasalahan penjadwalan *shift* jaga dokter di IGD, siklus algoritme, siklus penyelesaian menggunakan algoritme genetika dalam permasalahan optimasi penjadwalan *shift* jaga dokter di IGD, serta perancangan antar muka atau *user interface*.

4.1 Deskripsi Masalah

IGD (Instalasi Gawat Darurat) adalah unit dalam sebuah rumah sakit yang terdepan dalam menerima pasien saat pasien mengalami sebuah keadaan darurat (Jamil, 2015). Dibutuhkan banyak tenaga medis baik dokter maupun perawat untuk membantu perawatan pasien di dalam IGD. Agar pelayanan dokter di dalam IGD menjadi maksimal, maka diperlukan pembagian *shift* jaga di dalam IGD. Pada sub bab ini akan dibahas tentang penyelesaian permasalahan penjadwalan *shift* jaga dokter pada IGD. Dalam permasalahan ini, jadwal *shift* jaga dokter akan dioptimasi dengan menggunakan algoritme genetika. Perlunya pengoptimasian jadwal *shift* jaga dokter dikarenakan jadwal yang dibuat masih menggunakan sistem manual yang akan memakan waktu pembuatan serta dikhawatirkan jadwal yang dibuat terdapat kesalahan yang menyebabkan kurang adilnya jadwal *shift* jaga pada dokter di IGD. maka berdasarkan permasalahan tersebut diperlukan adanya sistem optimasi penjadwalan *shift* jaga dokter di IGD menggunakan algoritme genetika. Data yang digunakan untuk permasalahan optimasi penjadwalan *shift* jaga dokter di IGD ini adalah data yang diperoleh dari salah satu IGD di Rumah Sakit swasta di Malang.

Di dalam jadwal *shift* jaga dokter IGD yang ada, terdapat 11 dokter yang mendapatkan jadwal jaga. Jadwal *shift* jaga dokter tersebut terdiri dari 3 *shift. Shift* pertama pukul 07.00-14.00, *shift* kedua pukul 14.00-21.00, dan *shift* ketiga pukul 21.00-07.00. Tabel representasi untuk nama dokter beserta *ID*-nya dan tabel jadwal *shift* jaga dokter IGD akan ditunjukkan di Tabel 4.1 dan Tabel 4.2.

Tabel 4.1 Nama dan *ID* Dokter

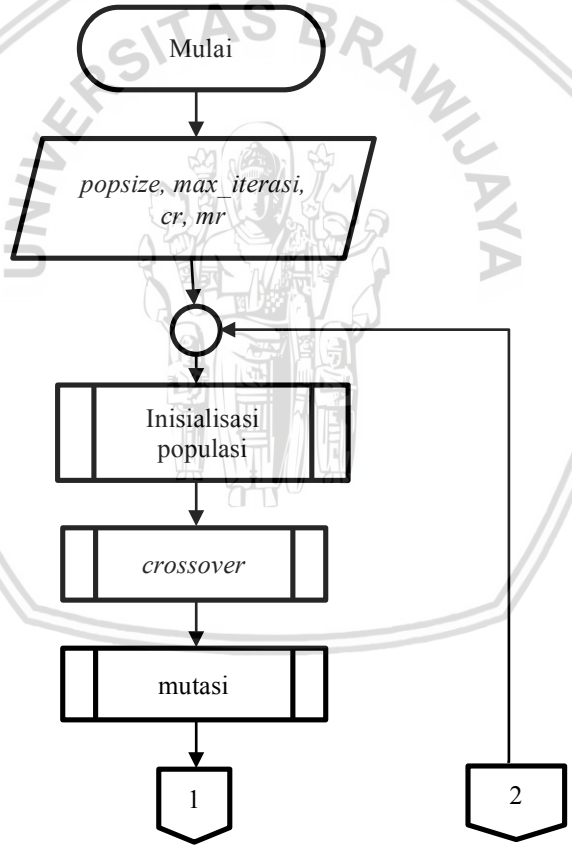
NAMA	ID
Dr. R	1
Dr. G	2
Dr. VHP	3
Dr. S	4
Dr. Z	5
Dr. Hjt	6
Dr. AR	7
Dr. MD	8
Dr. DI	9
Dr. DE	10
ISHIP	11

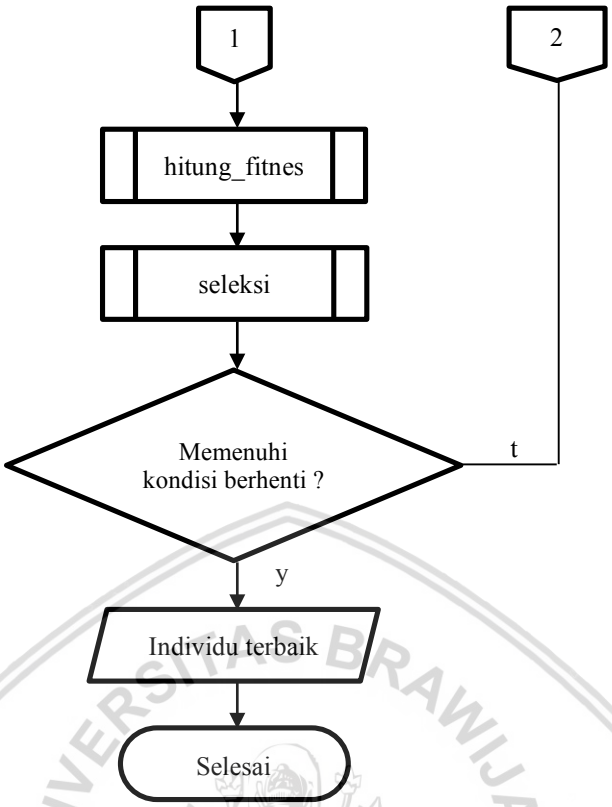
Tabel 4.2 Jadwal *Shift* Jaga Dokter di *IGD* Selama 3 Hari

Hari ke-	*Shift 1*				*Shift 2*			*Shift 3*
	PU	IGD	IGD	BPJS	PU	IGD	IGD	IGD
1	3	7	0	0	5	2	0	11
2	1	2	5	0	11	7	9	4
3	6	2	10	4	8	3	0	6

4.2 Siklus Algoritme

Algoritme genetika merupakan algoritme yang dapat menyelesaikan permasalahan yang cukup rumit jika diselesaikan secara manual. Algoritme ini merupakan salah satu algoritme yang digunakan untuk optimasi. Pada Gambar 4.1 akan digambarkan tahapan-tahapan proses penyelesaian pada algoritme genetika (Mahmudy, 2013).

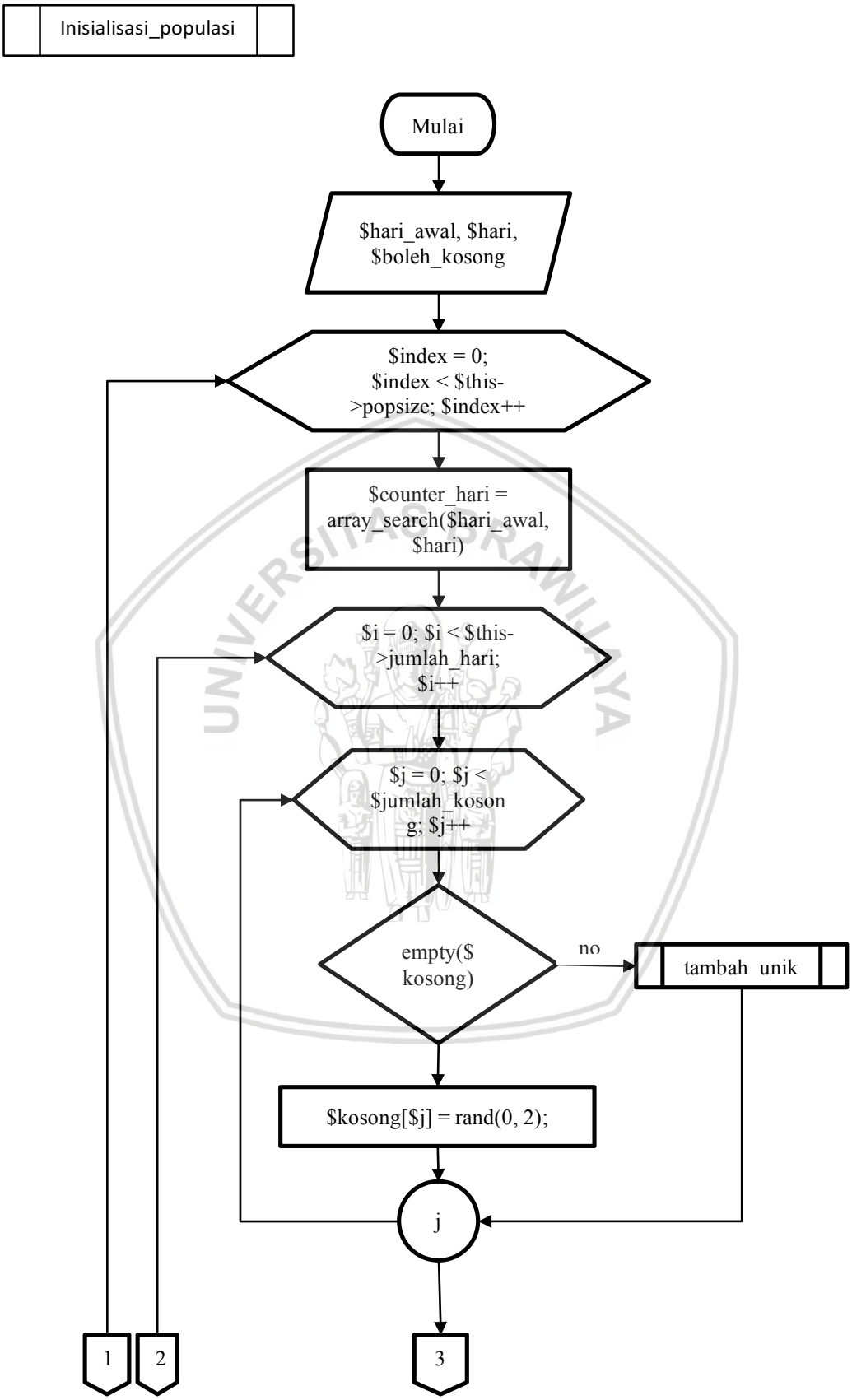


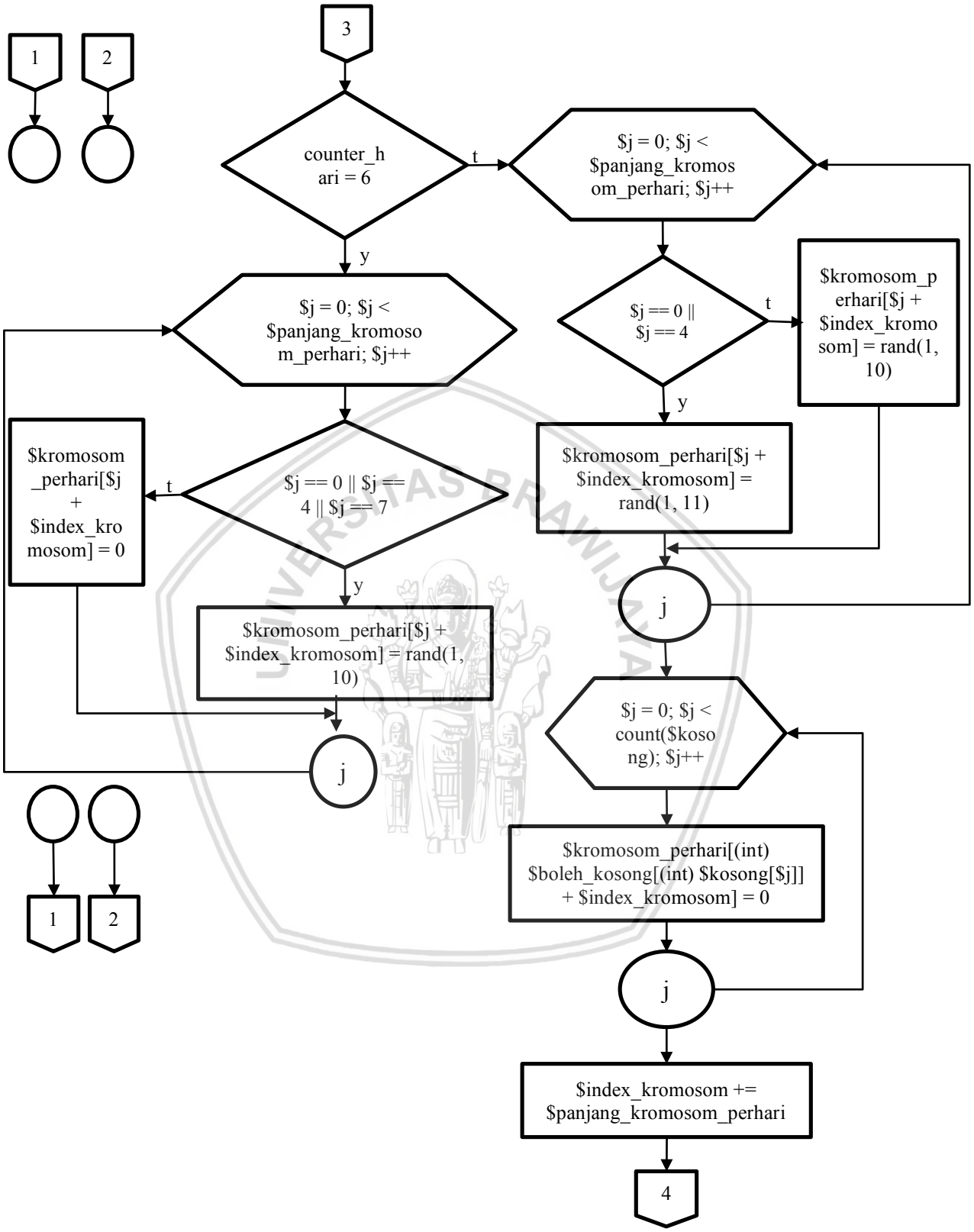


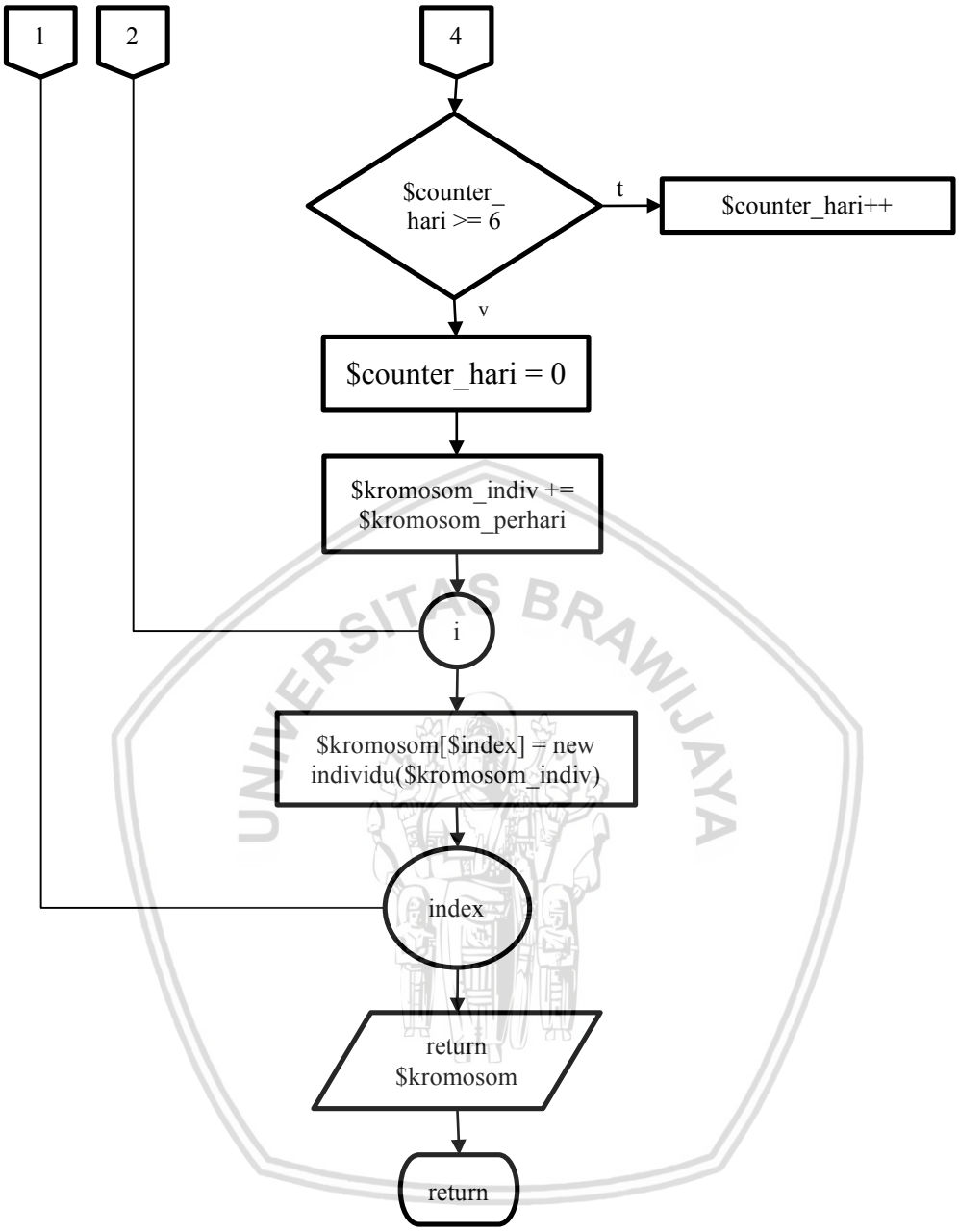
Gambar 4.1 *Flowchart* Siklus Algoritme Genetika

4.2.1 Inisialisasi Populasi Awal

Di dalam inisialisasi populasi, yang perlu diinisialisasi yaitu hari_awal (untuk menentukan hari pertama dalam 1 bulan), hari (untuk menentukan hari), dan boleh_kosong (untuk menentukan indeks yang kosong). Inisialisasi populasi merupakan pembangkitan populasi awal sebanyak jumlah populasi (*PopSize*) yang telah diinisialisasi sebelumnya. Representasi individu yang dilakukan menggunakan representasi kromosom dimana digunakan bilangan *integer* untuk representasinya. Perancangan untuk proses inisialisasi populasi awal akan digambarkan dalam diagram alir pada Gambar 4.2.

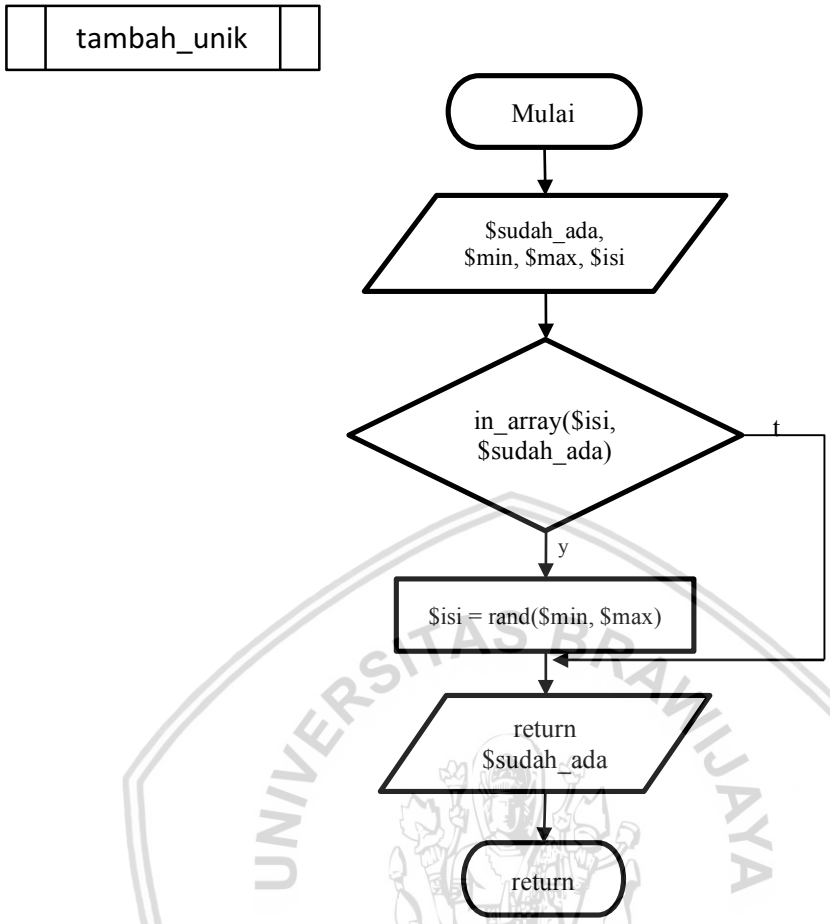

Inisialisasi_populasi
Mulai
$hari_awal, $hari, $boleh_kosong
$index = 0; $index < $this->popsize; $index++
$counter_hari = array_search($hari_awal, $hari)
$i = 0; $i < $this->jumlah_hari; $i++
$j = 0; $j < $jumlah_kosong; $j++
empty($kosong)
no
tambah_unik
$kosong[$j] = rand(0, 2);
j
1
2
3


3
1
2
counter_h
ari = 6
t
$j = 0; $j <
$panjang_kromos
om_perhari; $j++
y
$j = 0; $j <
$panjang_kromoso
m_perhari; $j++
$j == 0 ||
$j == 4
t
$kromosom_p
erhari[$j +
$index_kromo
som] = rand(1,
10)
y
$kromosom
_perhari[$j
+
$index_kro
mosom] = 0
t
$j == 0 || $j ==
4 || $j == 7
$kromosom_perhari[$j +
$index_kromosom] =
rand(1, 11)
y
$kromosom_perhari[$j +
$index_kromosom] = rand(1,
10)
j
$j = 0; $j <
count($koso
ng); $j++
j
$kromosom_perhari[(int)
$boleh_kosong[(int) $kosong[$j]]
+ $index_kromosom] = 0
1
2
j
$index_kromosom +=
$panjang_kromosom_perhari
4



Gambar 4.2 Diagram Alir Inisialisasi Populasi

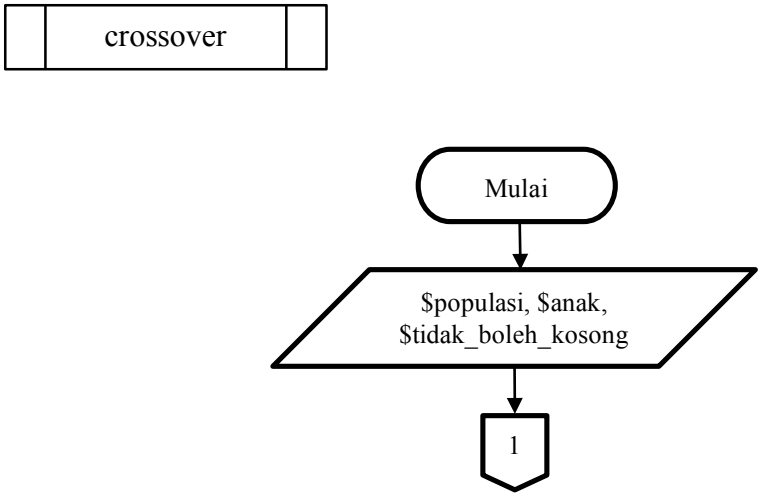
Kemudian seperti yang dijelaskan pada gambar diatas, terdapat *sub-function* yaitu fungsi tambah_unik. Fungsi ini digunakan untuk pencarian nilai acak pada fungsi boleh_kosong agar tidak terjadi penumpukan angka. Perancangan untuk proses tambah_unik akan digambarkan dalam diagram alir pada Gambar 4.3.

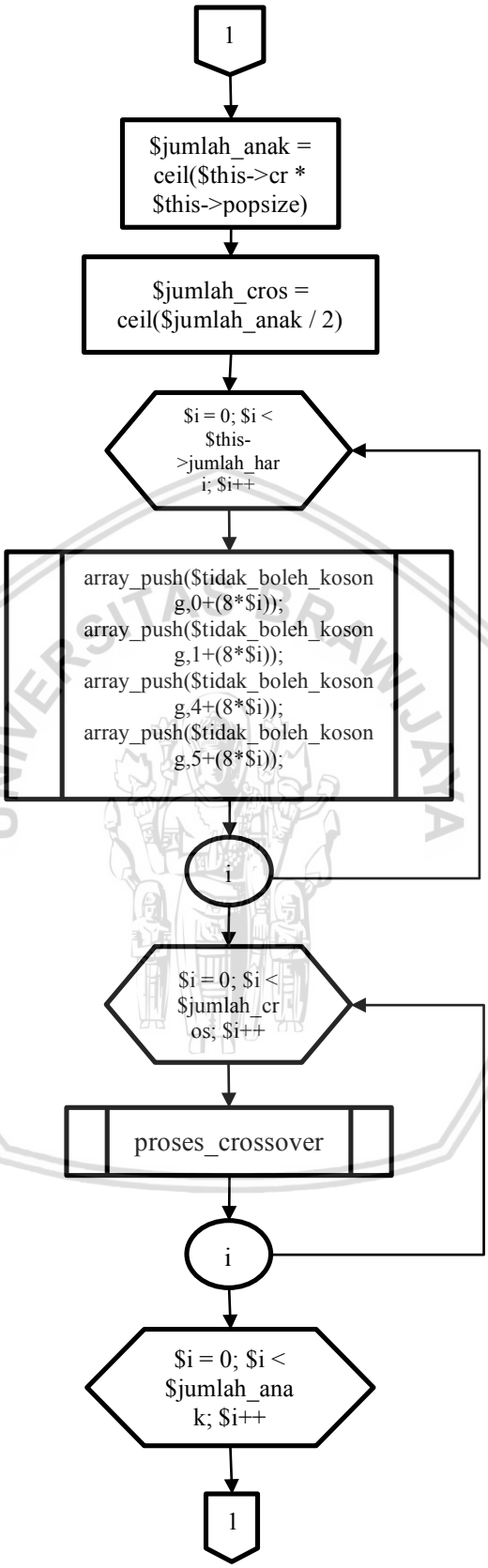


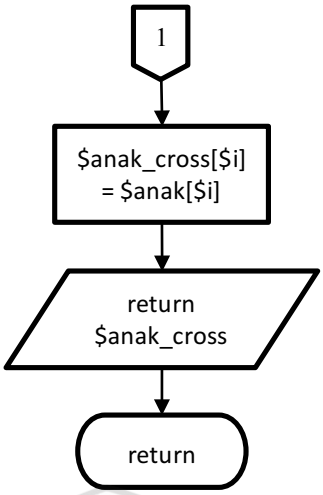
Gambar 4.3 Diagram Alir tambah_unik

4.2.2 Crossover

Melakukan proses reproduksi. Proses reproduksi yang pertama dilakukan yaitu menggunakan *crossover*. Metode *crossover* yang digunakan dalam penyelesaian penjadwalan *shift* jaga ini adalah *extended intermediate crossover*. Perancangan untuk proses reproduksi *crossover* akan digambarkan dalam diagram alir pada Gambar 4.4.

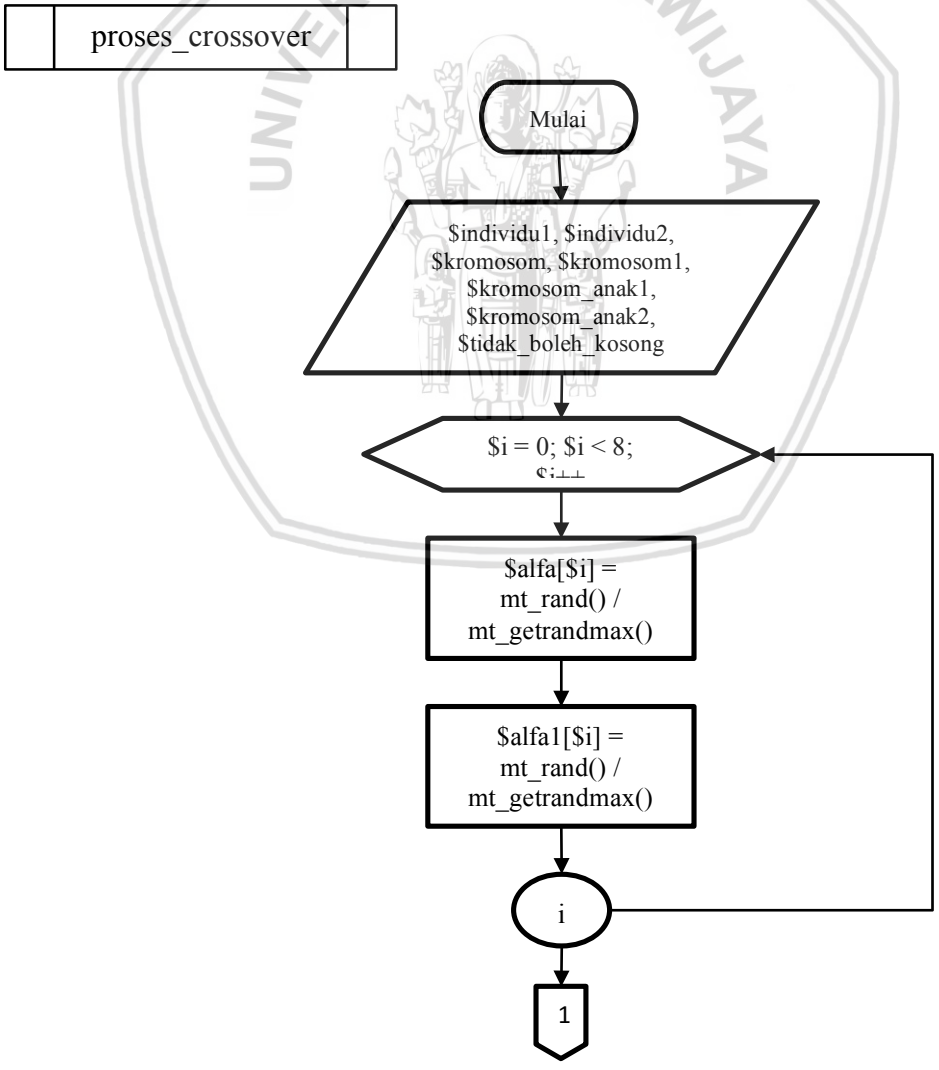


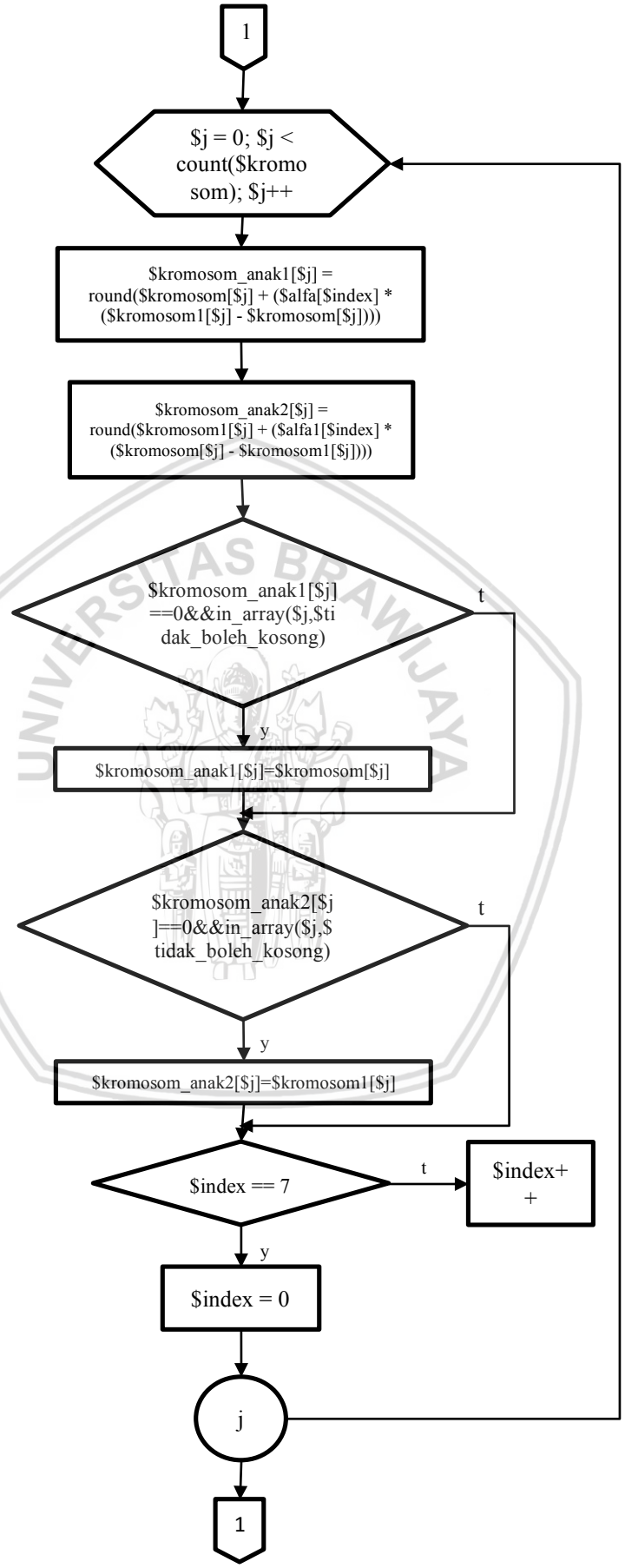

1
$jumlah_anak = ceil($this->cr * $this->popsize)
$jumlah_cros = ceil($jumlah_anak / 2)
$i = 0; $i < $this->jumlah_hari; $i++
array_push($tidak_boleh_kosong,0+(8*$i));
array_push($tidak_boleh_kosong,1+(8*$i));
array_push($tidak_boleh_kosong,4+(8*$i));
array_push($tidak_boleh_kosong,5+(8*$i));
i
$i = 0; $i < $jumlah_cros; $i++
proses_crossover
i
$i = 0; $i < $jumlah_anak; $i++
1

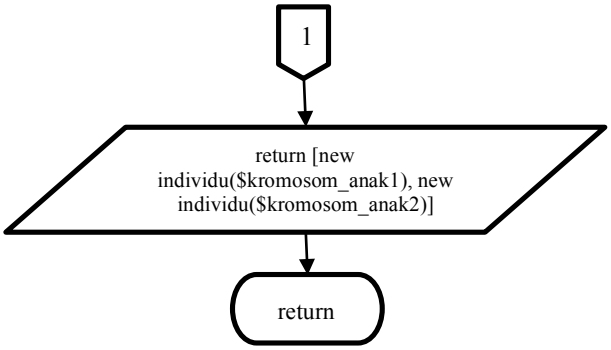


Gambar 4.4 Diagram Alir *Crossover*

Kemudian seperti yang dijelaskan pada gambar diatas, terdapat *sub-function* yaitu fungsi proses_crossover. Fungsi ini digunakan untuk melakukan proses reproduksi *crossover*. Perancangan untuk proses_crossover akan digambarkan dalam diagram alir pada Gambar 4.5.



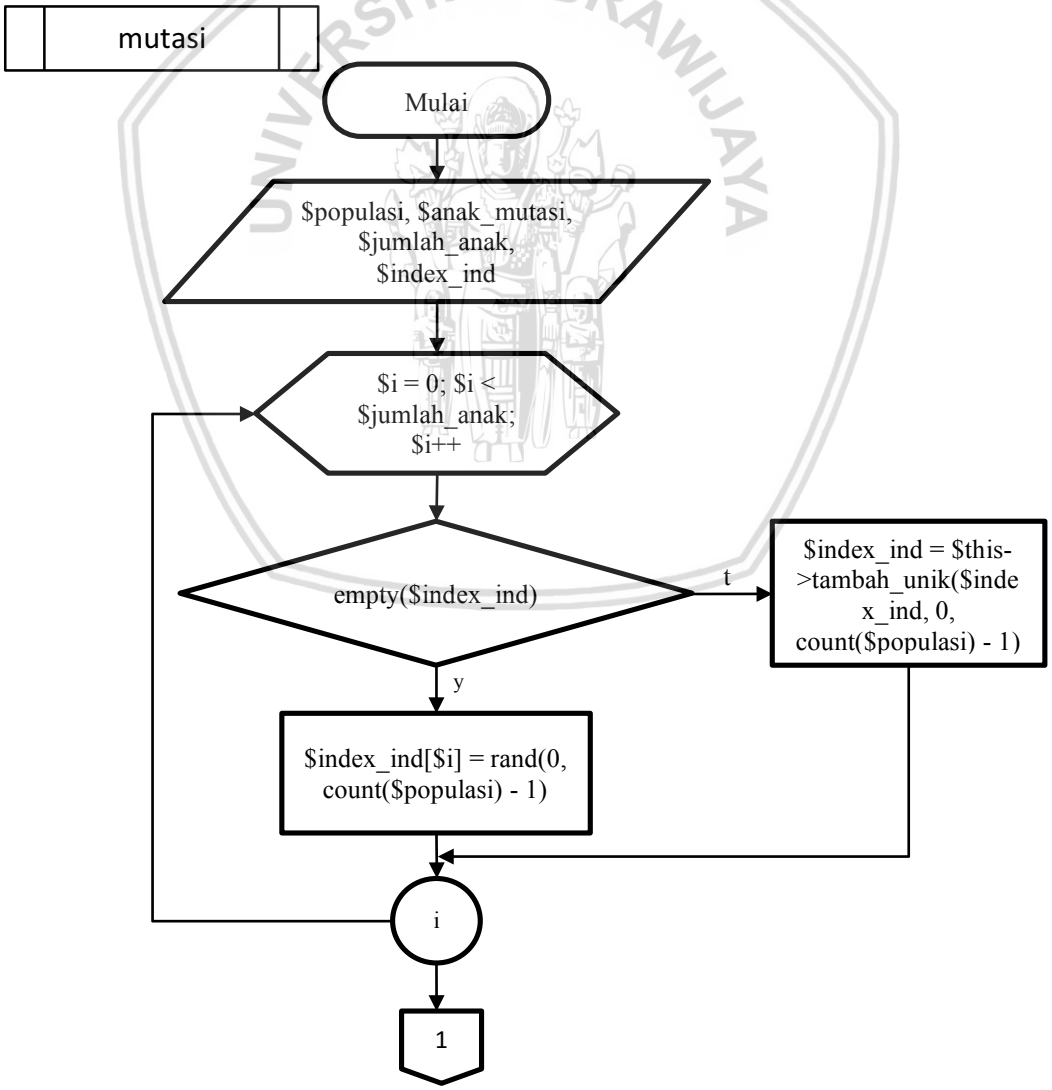

1
$j = 0; $j < count($kromosom); $j++
$kromosom_anak1[$j] = round($kromosom[$j] + ($alfa[$index] * ($kromosom1[$j] - $kromosom[$j])))
$kromosom_anak2[$j] = round($kromosom1[$j] + ($alfa1[$index] * ($kromosom[$j] - $kromosom1[$j])))
$kromosom_anak1[$j] ==0&&in_array($j,$tidak_boleh_kosong)
t
y
$kromosom_anak1[$j]=$kromosom[$j]
$kromosom_anak2[$j]==0&&in_array($j,$tidak_boleh_kosong)
t
y
$kromosom_anak2[$j]=$kromosom1[$j]
$index == 7
t
$index++
y
$index = 0
j
1

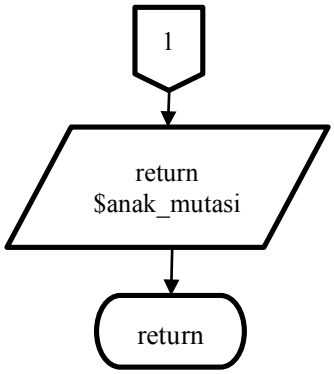


Gambar 4.5 Diagram Alir proses_crossover

4.2.3 Mutasi

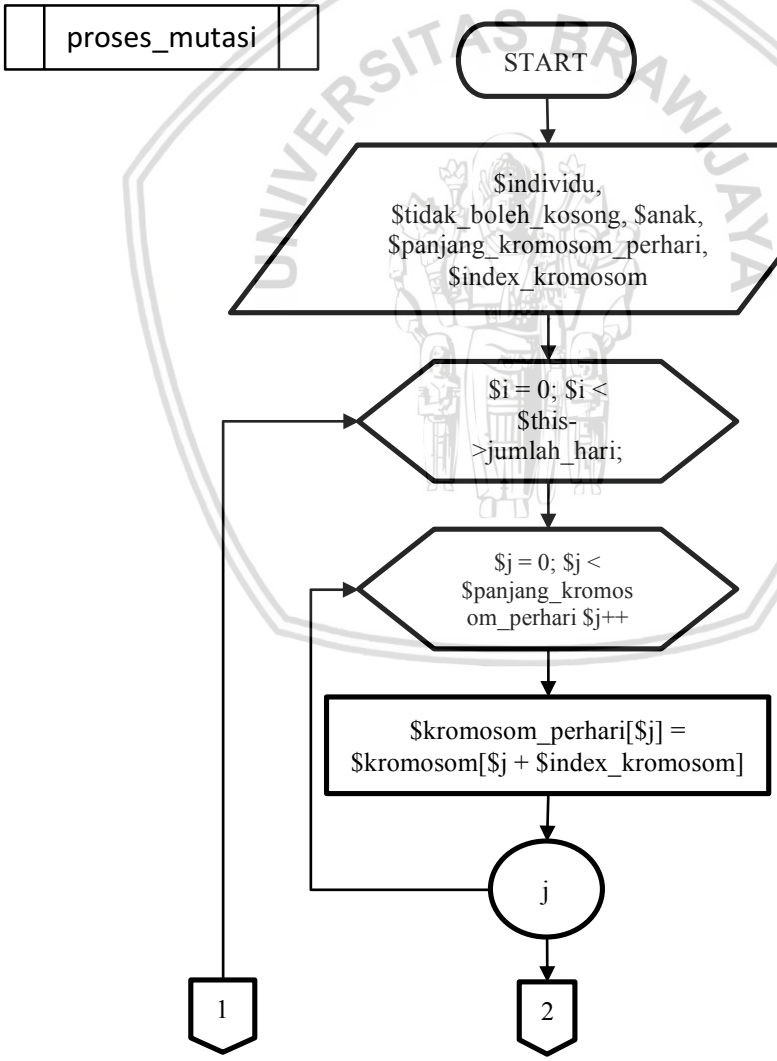
Melakukan proses reproduksi. Proses reproduksi yang dilakukan setelah metode *crossover* yaitu menggunakan mutasi. Metode mutasi yang digunakan dalam penyelesaian penjadwalan *shift* jaga ini adalah *reciprocal exchange mutation*. Perancangan untuk proses reproduksi mutasi akan digambarkan dalam diagram alir pada Gambar 4.6.

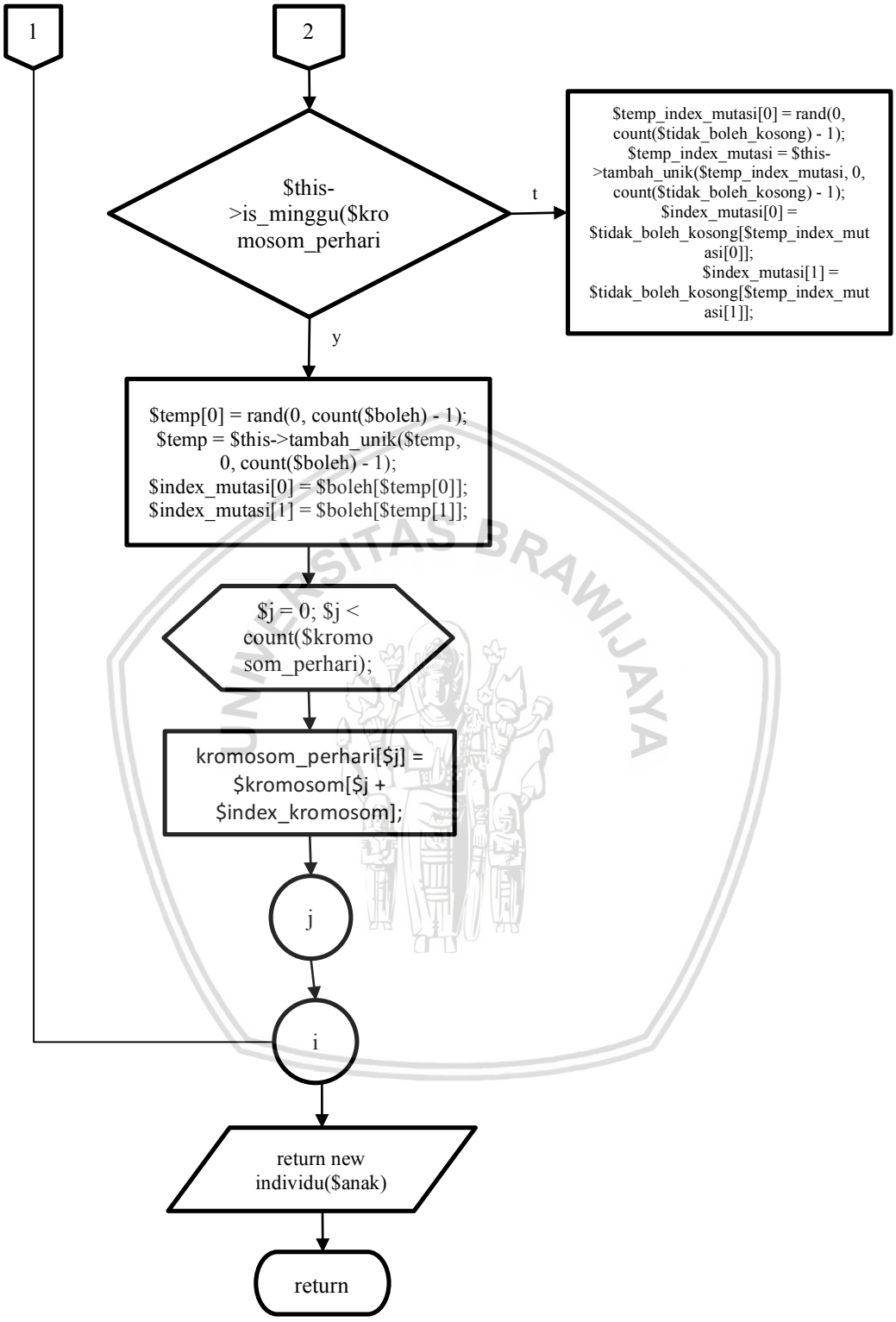




Gambar 4.6 Diagram Alir Mutasi

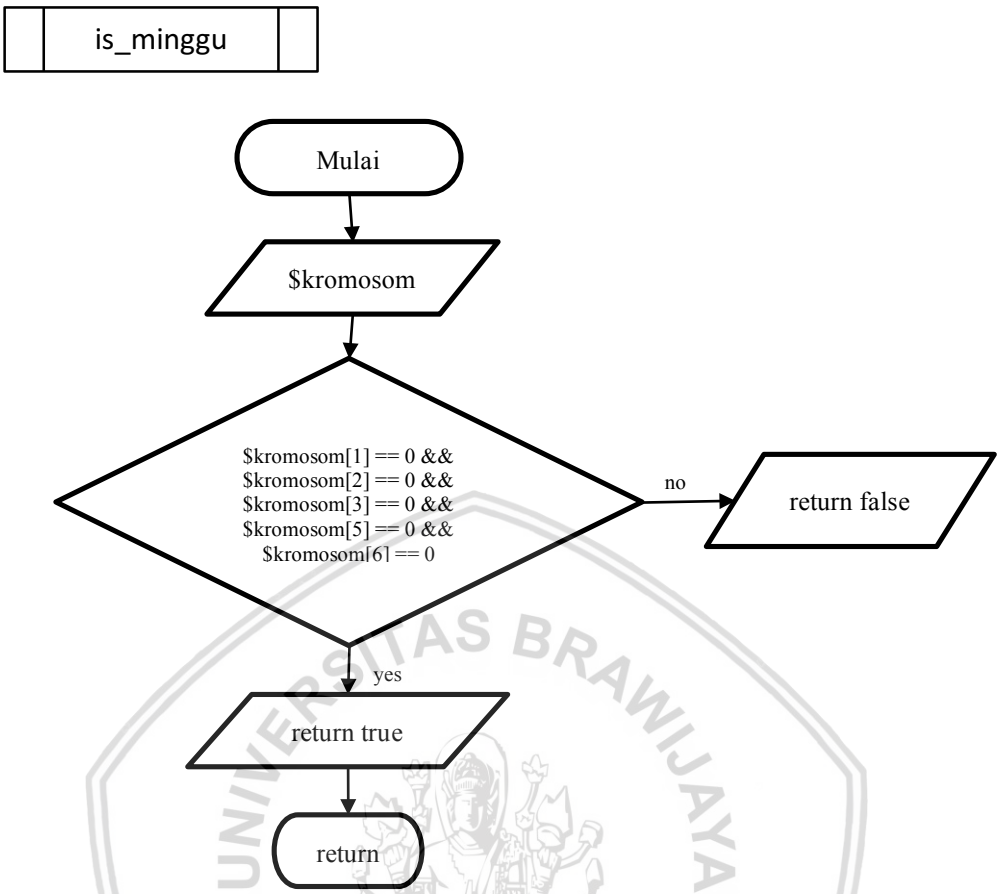
Kemudian seperti yang dijelaskan pada gambar diatas, terdapat *sub-function* yaitu fungsi proses_mutasi. Fungsi ini digunakan untuk melakukan proses reproduksi mutasi. Perancangan untuk proses_mutasi akan digambarkan dalam diagram alir pada Gambar 4.7.





Gambar 4.7 Diagram Alir proses_mutasi

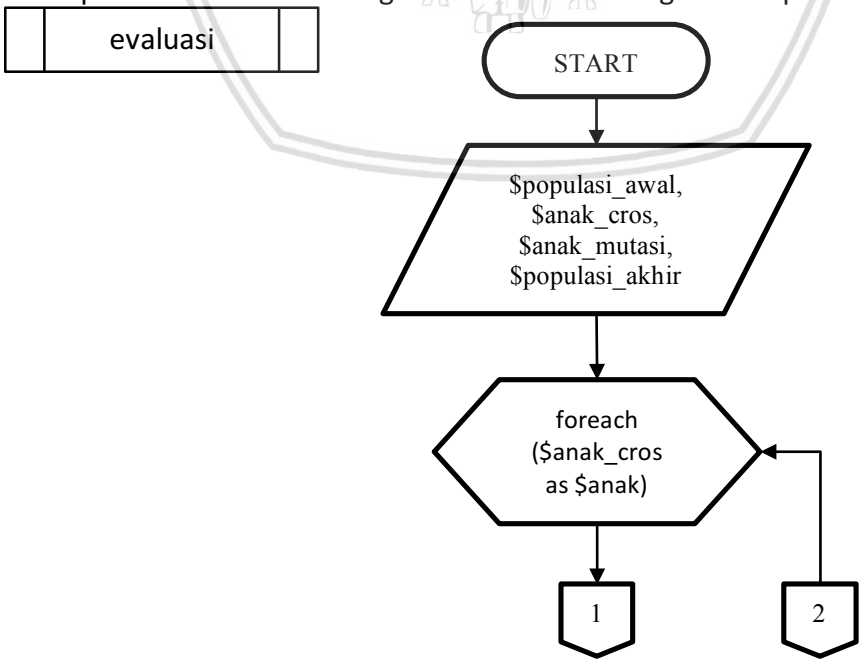
Kemudian seperti yang dijelaskan pada gambar diatas, terdapat *sub-function* yaitu fungsi is_minggu. Fungsi ini digunakan untuk menentukan bahwa hari tersebut merupakan hari minggu. Perancangan untuk is_minggu akan digambarkan dalam diagram alir pada Gambar 4.8.

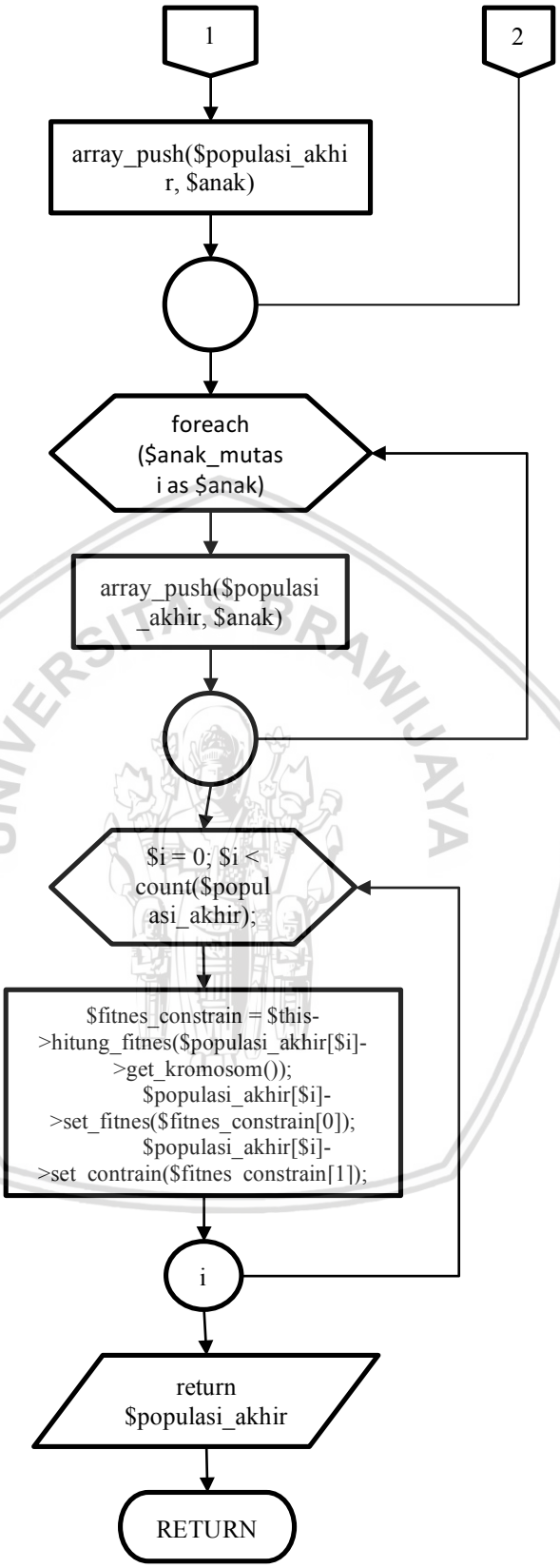


Gambar 4.8 Diagram Alir is_minggu

4.2.4 Evaluasi

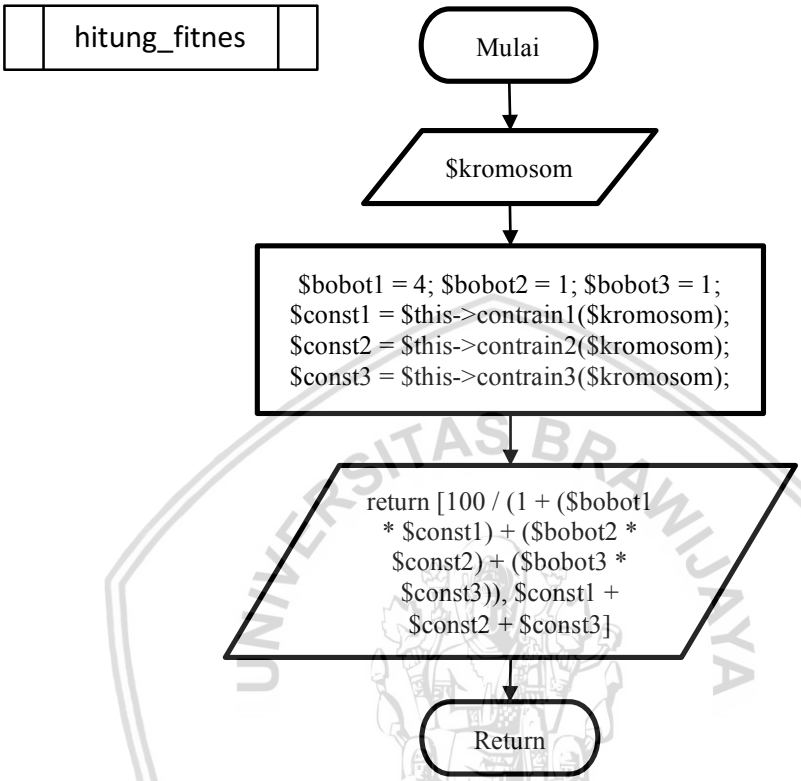
Melakukan evaluasi dengan cara menghitung nilai *fitness*. Perancangan untuk proses evaluasi akan digambarkan dalam diagram alir pada Gambar 4.9.





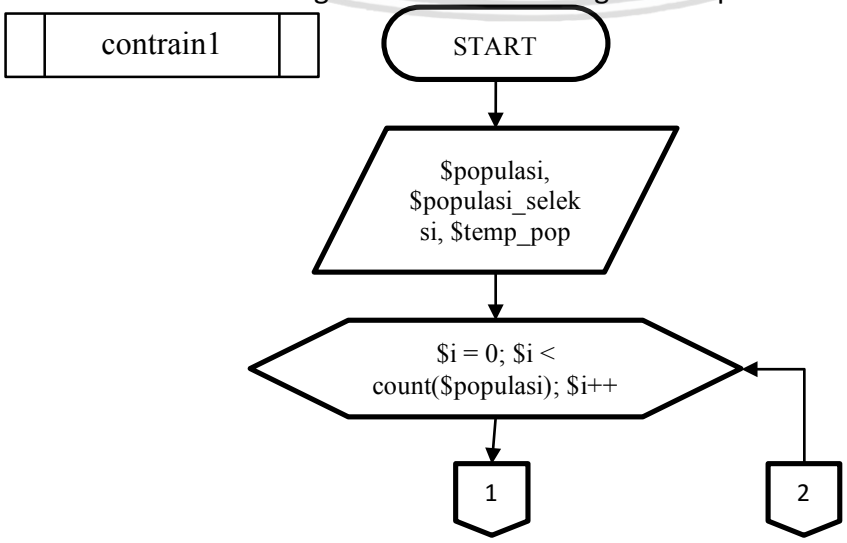
Gambar 4.8 Diagram Alir evaluasi

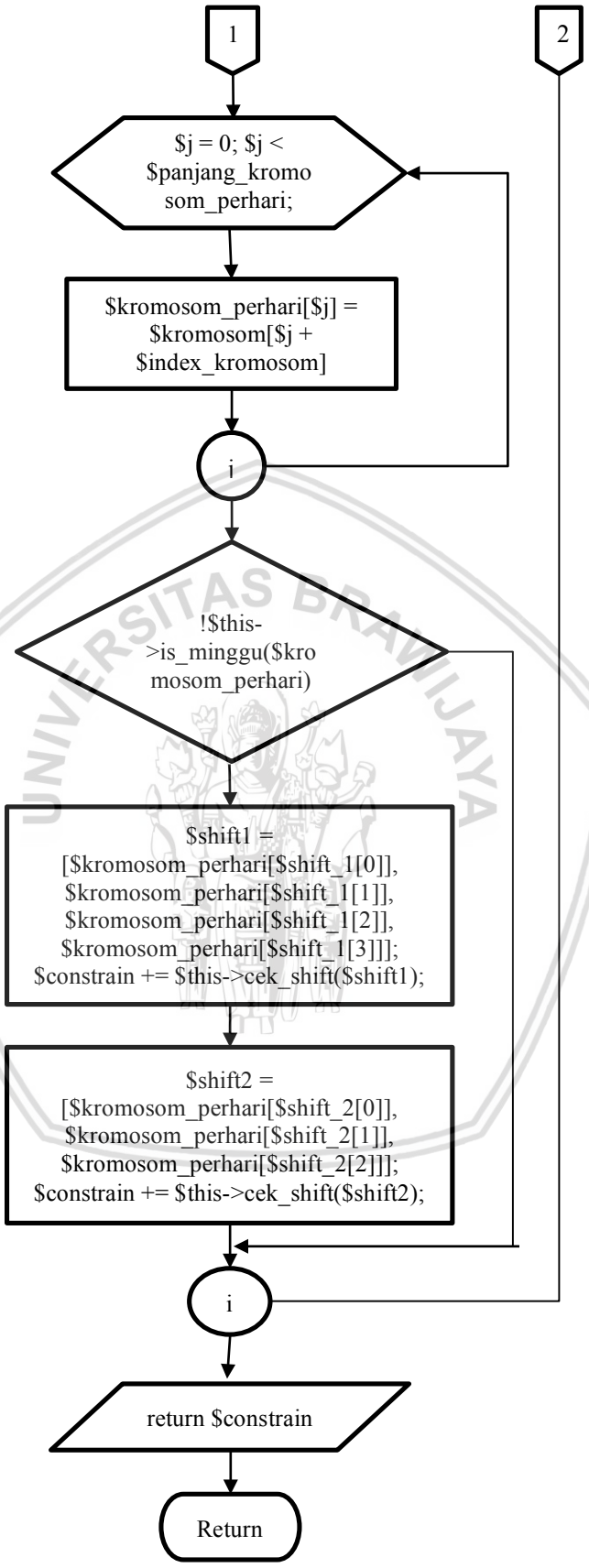
Kemudian seperti yang dijelaskan pada gambar diatas, terdapat *sub-function* yaitu fungsi hitung_fitnes. Fungsi ini digunakan untuk menentukan perhitungan nilai *fitness* serta penentuan bobot dari tiap *constraint* yang telah ditentukan sebelumnya. Perancangan untuk hitung_fitnes akan digambarkan dalam diagram alir pada Gambar 4.9.



Gambar 4.9 Diagram Alir hitung_fitnes

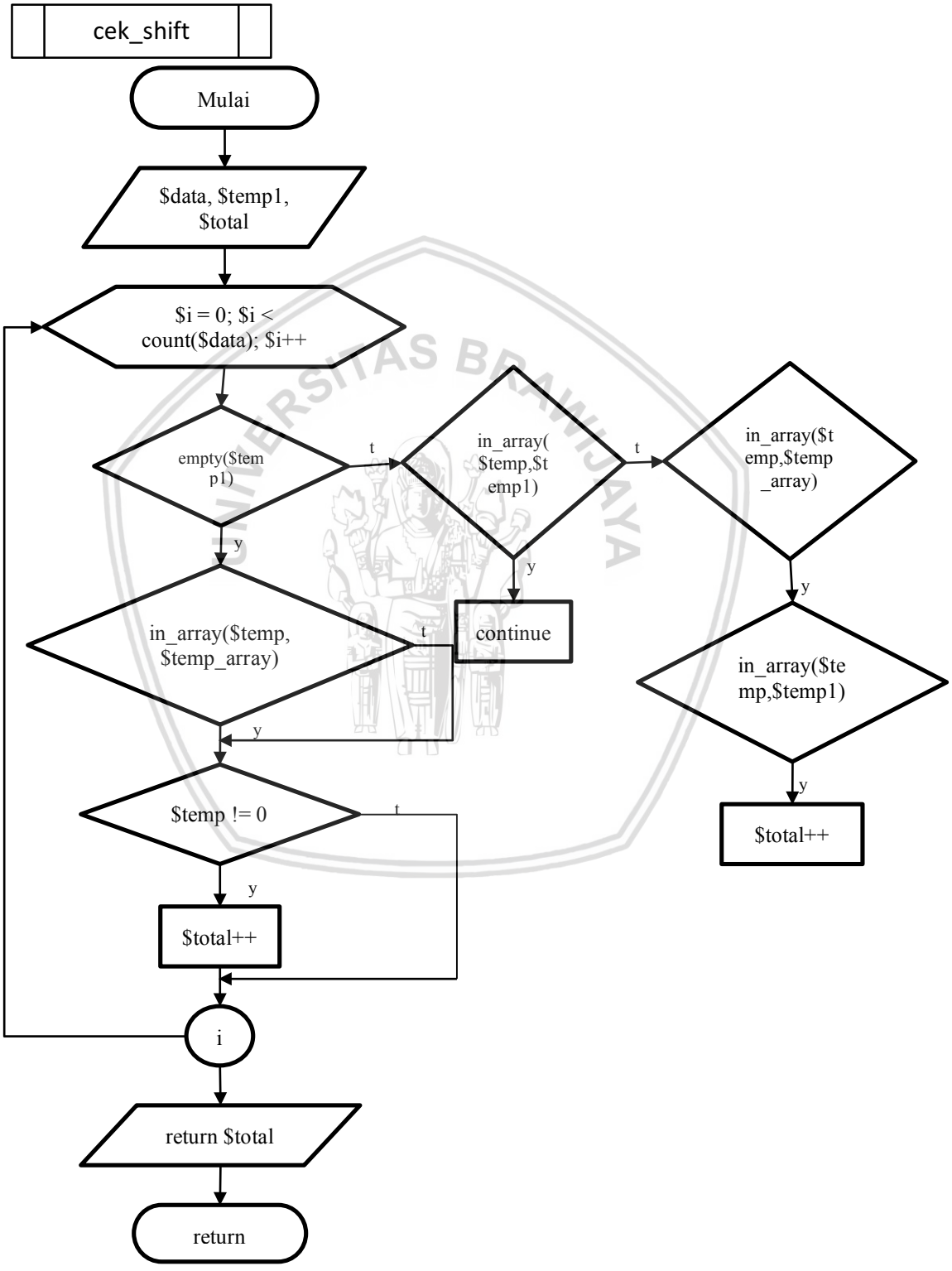
Kemudian seperti yang dijelaskan pada gambar diatas, terdapat *sub-function* yaitu fungsi contrain1. Fungsi ini digunakan untuk perhitungan nilai *fitness* pada *constrain* 1. *Constrain* 1 ini merupakan *hard constraint* yang memiliki aturan bahwa setiap *shift* tidak boleh terdapat 2 nama/*ID* dokter yang sama. Perancangan untuk contrain1 akan digambarkan dalam diagram alir pada Gambar 4.10.





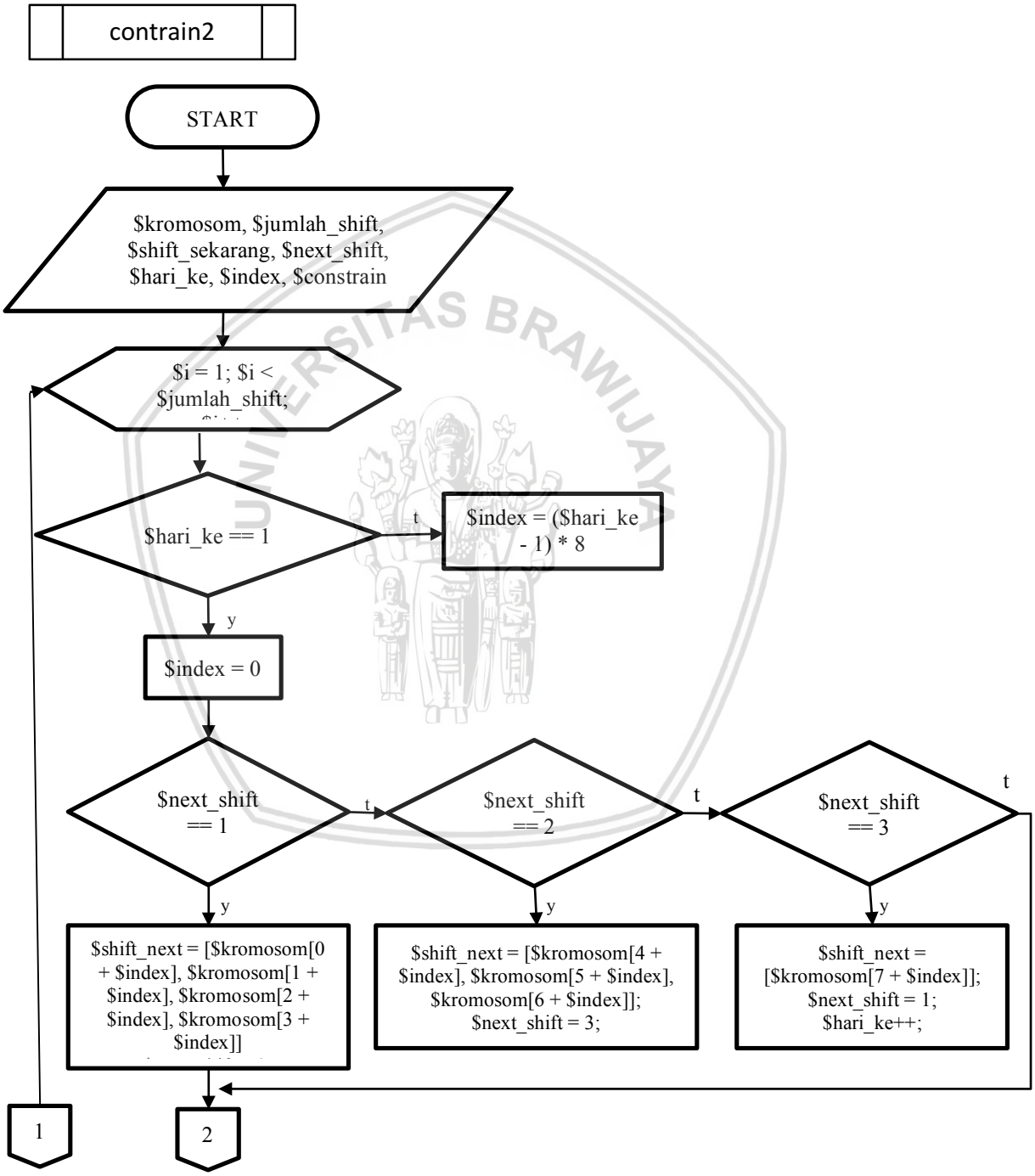
Gambar 4.10 Diagram Alir contrain1

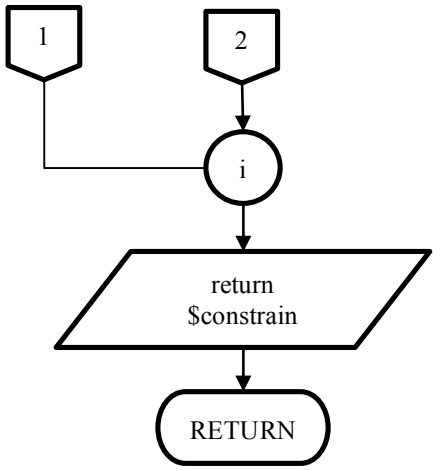
Kemudian seperti yang dijelaskan pada gambar diatas, terdapat *sub-function* yaitu fungsi cek_shift. Fungsi ini digunakan untuk melakukan pemeriksaan terhadap ada tidaknya data yang terkena *constrain* 1. Perancangan untuk cek_shift akan digambarkan dalam diagram alir pada Gambar 4.11.



Gambar 4.11 Diagram Alir cek_shift

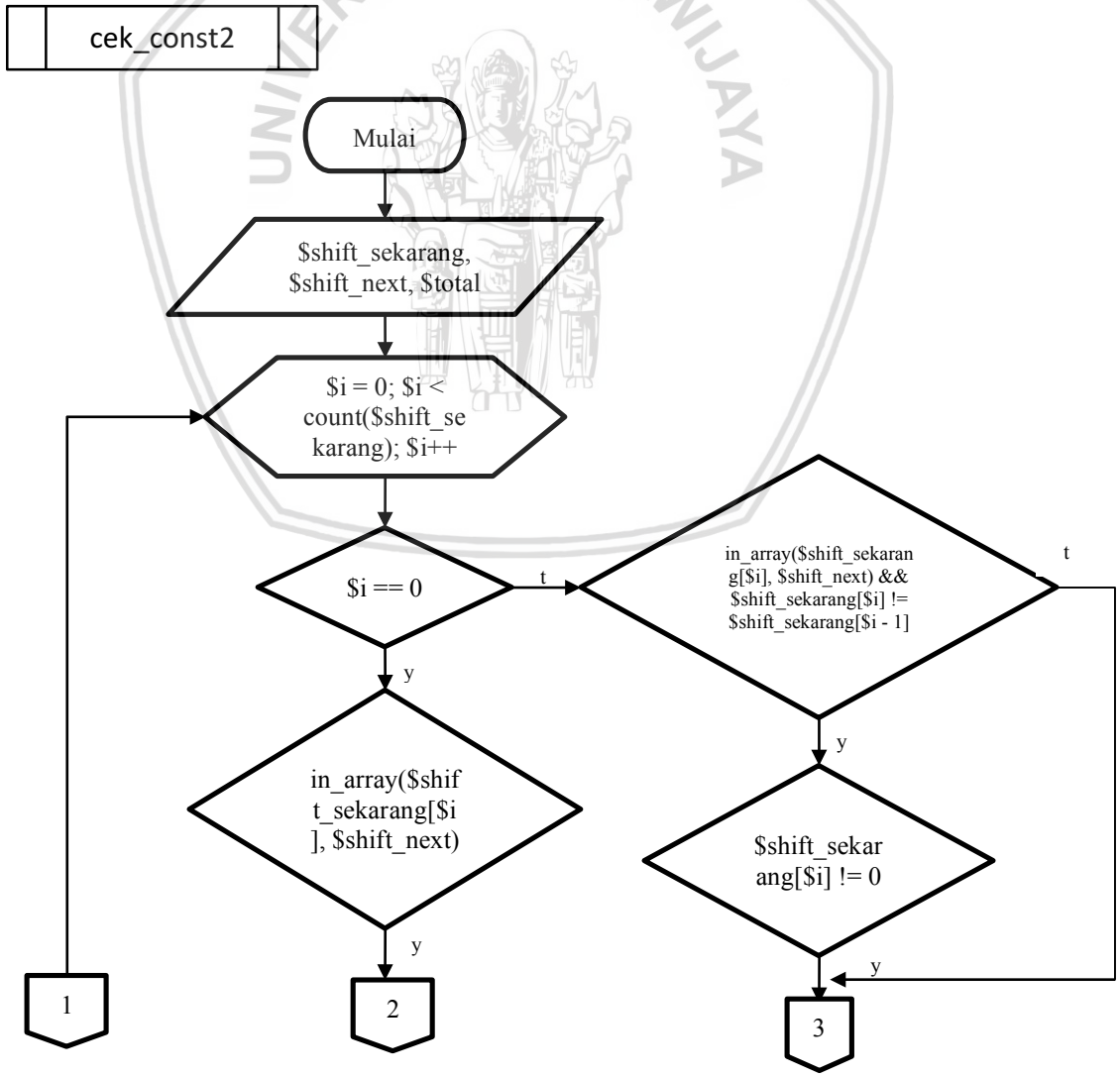
Kemudian seperti yang dijelaskan pada Gambar 4.8, terdapat *sub-function* yaitu fungsi contrain2. Fungsi ini digunakan untuk perhitungan nilai *fitness* pada *constrain* 2. *Constrain* 2 ini merupakan *soft* constraint memiliki aturan bahwa setiap nama/*ID* dokter tidak boleh bekerja dengan *shift* yang berurutan. Perancangan untuk contrain2 akan digambarkan dalam diagram alir pada Gambar 4.12.

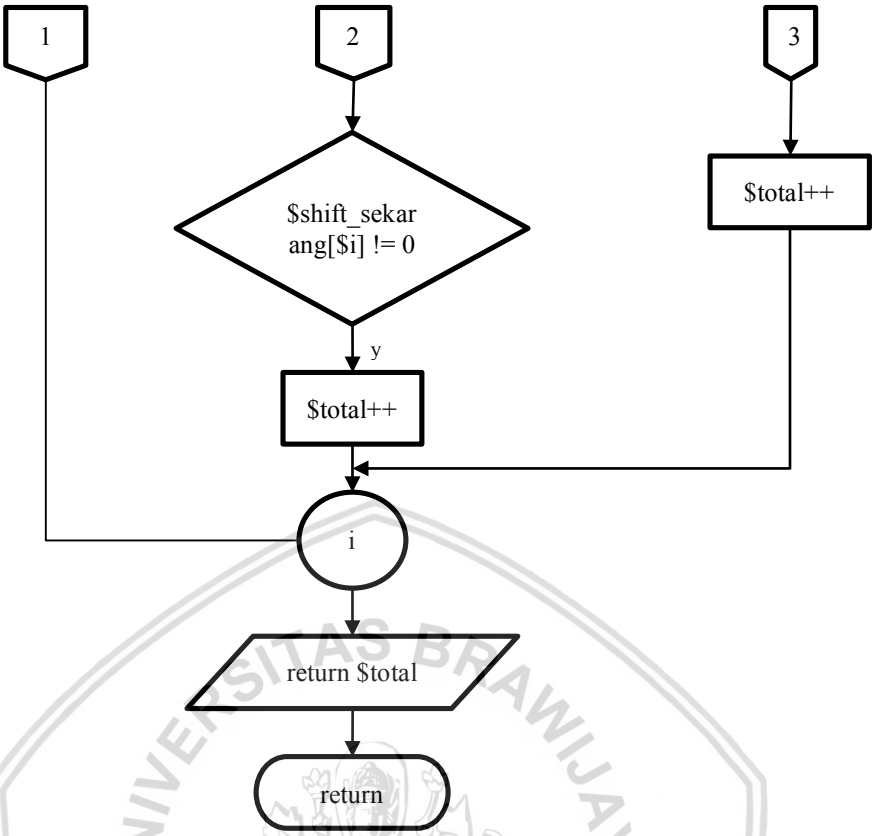




Gambar 4.12 Diagram Alir contrain2

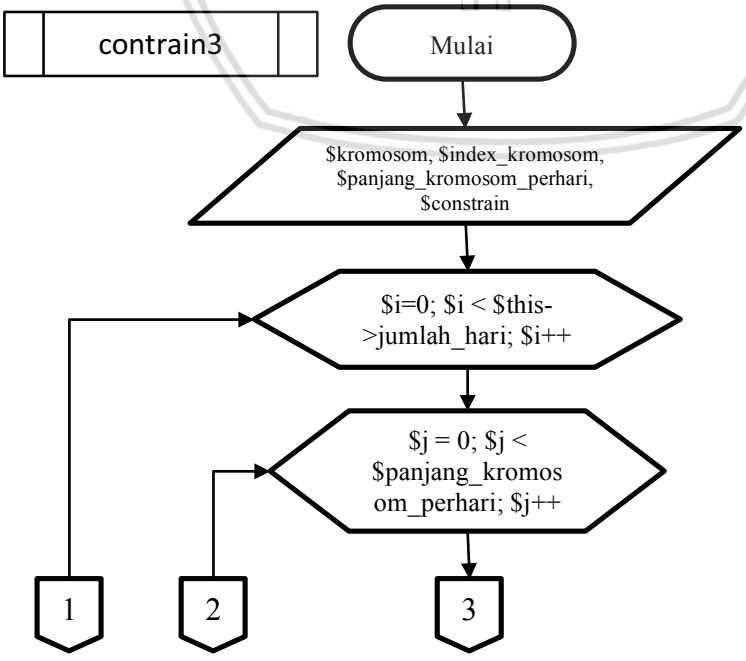
Kemudian seperti yang dijelaskan pada gambar diatas, terdapat *sub-function* yaitu fungsi cek_const2. Fungsi ini digunakan untuk melakukan pemeriksaan terhadap ada tidaknya data yang terkena *constrain* 2. Perancangan untuk cek_const2 akan digambarkan dalam diagram alir pada Gambar 4.13.

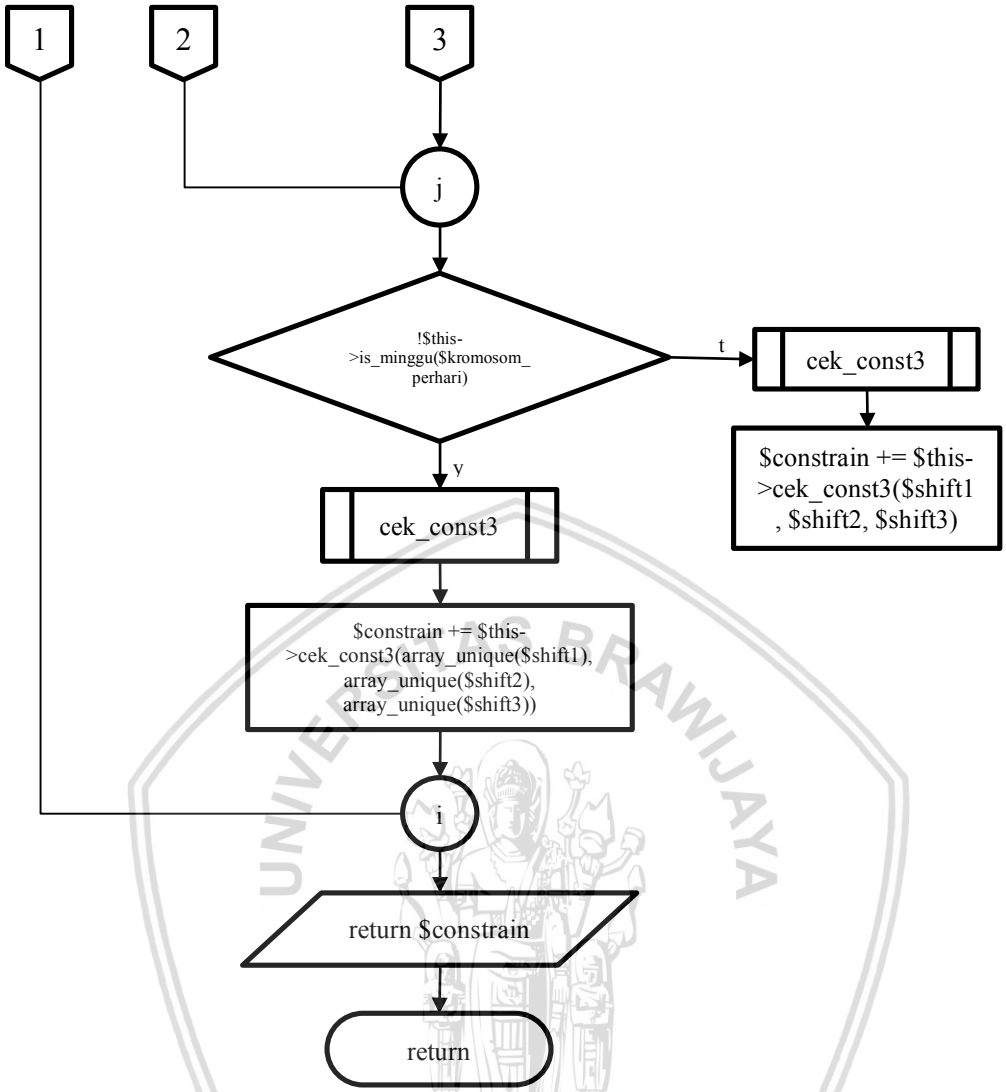




Gambar 4.13 Diagram Alir cek_const2

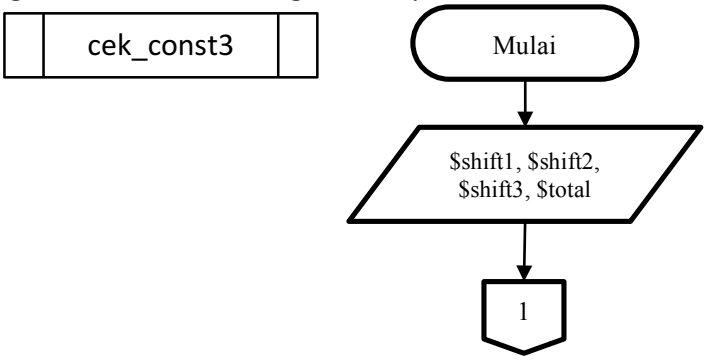
Kemudian seperti yang dijelaskan pada Gambar 4.8, terdapat *sub-function* yaitu fungsi contrain3. Fungsi ini digunakan untuk perhitungan nilai *fitness* pada *constrain* 3. *Constrain* 2 ini merupakan *soft* constraint memiliki aturan bahwa setiap nama/*ID* dokter tidak boleh bekerja lebih dari 1 *shift*. Perancangan untuk contrain2 akan digambarkan dalam diagram alir pada Gambar 4.14.

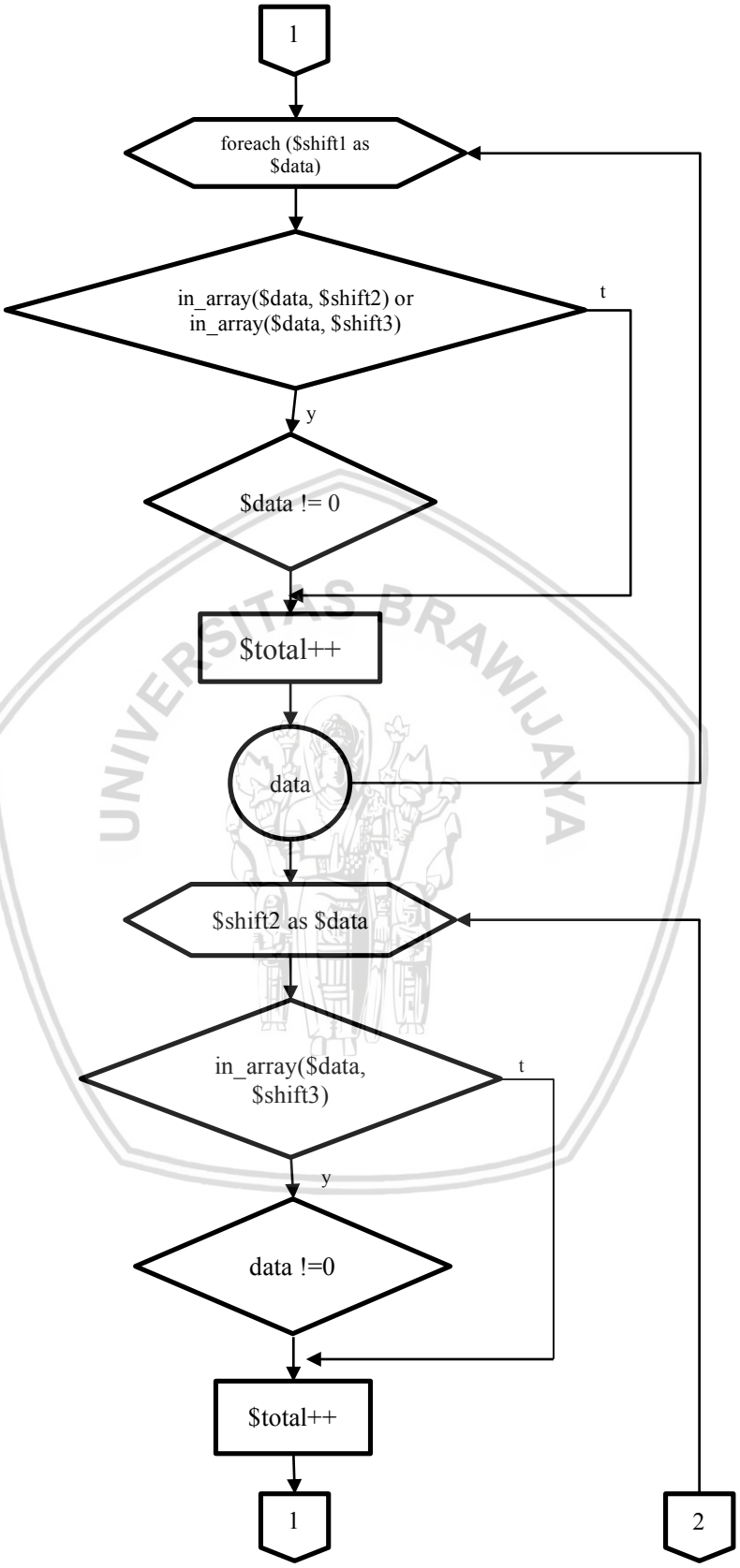


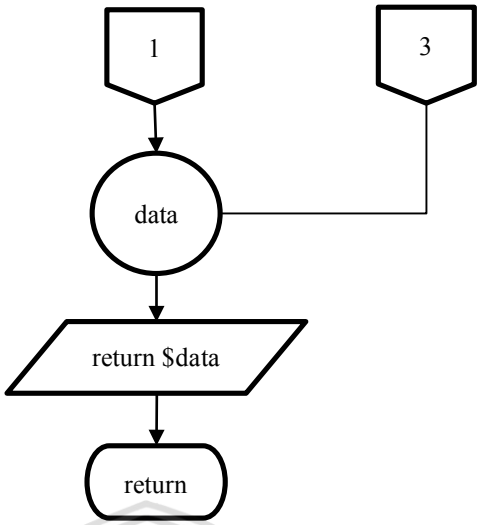


Gambar 4.13 Diagram Alir contrain3

Kemudian seperti yang dijelaskan pada gambar diatas, terdapat *sub-function* yaitu fungsi cek_const3. Fungsi ini digunakan untuk melakukan pemeriksaan terhadap ada tidaknya data yang terkena *constrain* 3. Perancangan untuk cek_const3 akan digambarkan dalam diagram alir pada Gambar 4.14.



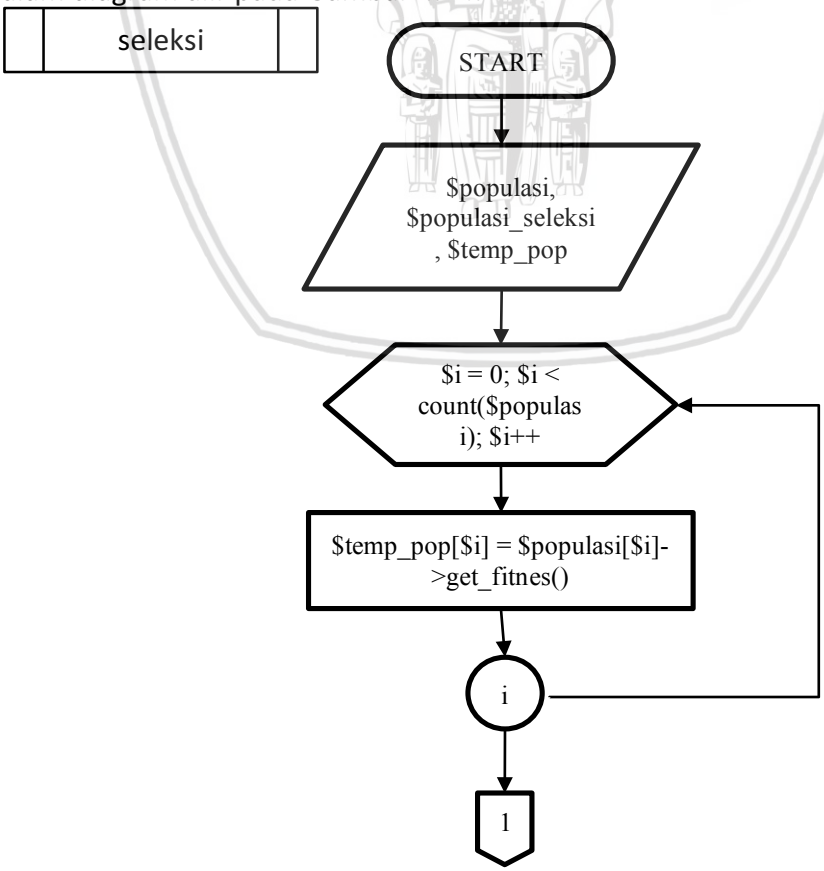

1
foreach ($shift1 as $data)
in_array($data, $shift2) or in_array($data, $shift3)
t
y
$data != 0
$total++
data
$shift2 as $data
in_array($data, $shift3)
t
y
data !=0
$total++
1
2

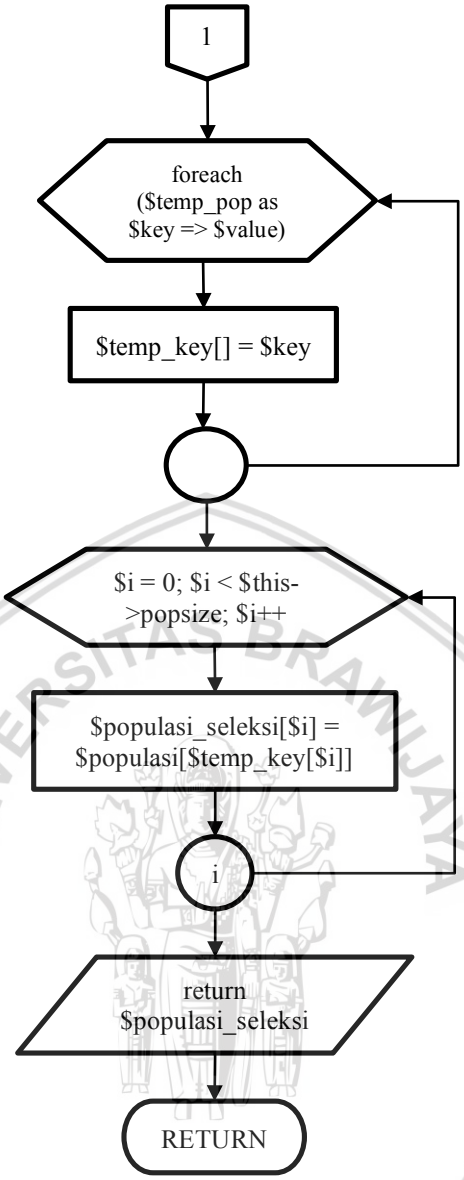


Gambar 4.13 Diagram Alir cek_contrain3

4.2.5 Seleksi

Melakukan seleksi dengan metode *elitism* yaitu memilih individu terbaik yang lolos untuk generasi selanjutnya. Iterasi akan dihentikan ketika memenuhi kondisi berhenti dan menghasilkan solusi terbaik dengan individu yang memiliki nilai *fitness* tertinggi. Namun apabila kriteria kondisi berhenti belum juga terpenuhi, maka proses iterasi akan terus dilanjutkan dan dimulai dari proses inisialisasi populasi kembali. Perancangan untuk proses seleksi akan digambarkan dalam diagram alir pada Gambar 4.14.





Gambar 4.14 Diagram Alir seleksi

4.3 Penyelesaian Masalah dengan Algoritme Genetika

Penyelesaian masalah dengan algoritme genetika dilakukan untuk mengoptimalkan penjadwalan *shift* jada yang ada di IGD. Di dalam penyelesaian ini akan ditentukan mulai dari inisialisasi, reproduksi, evaluasi, seleksi, hingga proses iterasi untuk mendapatkan hasil yang terbaik. Di dalam evaluasi akan dihitung *fitness*-nya menggunakan *soft constraint* dan *hard constraint* yang akan ditentukan nantinya.

4.3.1 Inisialisasi Populasi Awal

Di dalam inisialisasi populasi awal, akan dilakukan inisialisasi pada parameter berupa:

1. Jumlah populasi(*popSize*) : 3
2. Jumlah generasi : 2
3. *Crossover rate*/probabilitas *crossover* : 0.6

4. *Mutation rate*/probabilitas mutasi : 0.4

Pada tahap ini, populasi awal akan dibangkitkan sebanyak jumlah populasi yang telah ditentukan. Kemudian kromosom yang dibangkitkan terdiri dari 3 bagian yang masing-masing bagian diisi 4, 3, dan 1 angka representasi *ID* dokter. Jadi total jumlah kromosom adalah 3 shift yang diisikan 11 *ID* dokter dikalikan dengan 3 hari yang menghasilkan 24. Contoh bentuk 1 kromosom akan digambarkan pada Gambar 4.15.

Shift 1 PU	Shift 1 IGD	Shift 1 IGD	Shift 1 BPJS	Shift 2 PU	Shift 2 IGD	Shift 2 IGD	Shift 3 IGD
1	2	0	0	5	2	0	5

Gambar 4.15 Bentuk Kromosom

Dan inisialisasi populasi awal dapat dilihat di Tabel 4.3.

Tabel 4.3 Inisialisasi Populasi Awal (*popSize*)

P1

Hari	Shift 1 PU	Shift 1 IGD	Shift 1 IGD	Shift 1 BPJS	Shift 2 PU	Shift 2 IGD	Shift 2 IGD	Shift 3 IGD
1	1	2	0	0	5	2	0	5
2	6	1	0	0	3	5	0	4
3	4	0	0	0	2	0	0	7

P2

Hari	Shift 1 PU	Shift 1 IGD	Shift 1 IGD	Shift 1 BPJS	Shift 2 PU	Shift 2 IGD	Shift 2 IGD	Shift 3 IGD
1	9	4	0	0	3	2	10	8
2	6	2	10	4	11	7	9	8
3	2	0	0	0	7	0	0	8

P3

Hari	Shift 1 PU	Shift 1 IGD	Shift 1 IGD	Shift 1 BPJS	Shift 2 PU	Shift 2 IGD	Shift 2 IGD	Shift 3 IGD
1	9	4	0	4	11	7	9	8
2	3	2	10	0	11	7	10	8
3	2	0	0	0	7	0	0	8

4.3.2 Reproduksi

Proses reproduksi yang dilakukan pada manualisasi untuk optimasi penjadwalan *shift* jaga dokter di IGD ini menggunakan 2 proses reproduksi. Yang pertama adalah proses *crossover* dan berikutnya adalah proses mutasi.

4.3.2.1 Crossover

Metode *crossover* yang digunakan adalah *extended intermediate crossover*. Digunakannya *extended intermediate crossover* pada penelitian ini karena nilai *offspring* yang dihasilkan berasal dari 2 kombinasi induk secara acak yang perhitungannya menggunakan nilai *alpha*. *Offspring* dapat dihasilkan bergantung pada nilai *crossover rate* dikalikan dengan jumlah popsize. Perhitungan jumlah *offspring* adalah 0.6 x 3 = 1.8 ≈ 2 maka *offspring*-nya sebanyak 2.

Tahapan dari *crossover* ini dimulai dengan pemilihan *parent* secara acak dari populasi yang ada. Misalnya *parent* yang dipilih adalah P1 dan P3. Kemudian seperti yang dijelaskan sebelumnya pada Persamaan 2.1 dan 2.2, perhitungan *crossover extended intermediate* ini menggunakan nilai *alpha* (α) yang dipilih secara acak pada interval [-0.25, 1.25]. Berikut merupakan nilai *alpha* yang ditetapkan secara acak sejak awal yang akan ditunjukkan pada Tabel 4.4.

Tabel 4.4 Penentuan Nilai *Alpha (α)*

P1

Nilai α	Indeks ke-1	Indeks ke-2	Indeks ke-3	Indeks ke-4	Indeks ke-5	Indeks ke-6	Indeks ke-7	Indeks ke-8
	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8

P2

Nilai α	Indeks ke-1	Indeks ke-2	Indeks ke-3	Indeks ke-4	Indeks ke-5	Indeks ke-6	Indeks ke-7	Indeks ke-8
	0.9	1	0.8	0.7	0.4	0.6	0.5	0.3

Setelah ditentukan *parent*-nya secara acak yaitu P1 dan P3, maka dilakukan perhitungan dengan menggunakan nilai *alpha* yang akan menghasilkan *offspring* sebanyak 2. Berikut adalah contoh perhitungan menggunakan *extended intermediate crossover*:

C1 : $x_1 = 1 + (0.1 * (9 - 1)) = 1.8 \approx 2$
$x_2 = 2 + (0.2 * (4 - 2)) = 2.4 \approx 2$
$x_3 = 0 + (0.3 * (0 - 0)) = 0$

C1 : $x_1 = 9 + (0.9 * (1 - 9)) = 1.8 \approx 2$
$x_2 = 4 + (1 * (2 - 4)) = 2$
$x_3 = 0 + (0.8 * (0 - 0)) = 0$

P1

Hari	Shift 1 PU	Shift 1 IGD	Shift 1 IGD	Shift 1 BPJS	Shift 2 PU	Shift 2 IGD	Shift 2 IGD	Shift 3 IGD
1	1	2	0	0	5	2	0	5
2	6	1	0	0	3	5	0	4
3	4	0	0	0	2	0	0	7

P3

Hari	Shift 1 PU	Shift 1 IGD	Shift 1 IGD	Shift 1 BPJS	Shift 2 PU	Shift 2 IGD	Shift 2 IGD	Shift 3 IGD
1	9	4	0	4	11	7	9	8
2	3	2	10	0	11	7	10	8
3	2	0	0	0	7	0	0	8

C1

Hari	Shift 1 PU	Shift 1 IGD	Shift 1 IGD	Shift 1 BPJS	Shift 2 PU	Shift 2 IGD	Shift 2 IGD	Shift 3 IGD
1	2	2	0	2	8	5	6	7
2	6	1	3	0	7	6	7	7
3	4	0	0	0	5	0	0	8

C2

Hari	Shift 1 PU	Shift 1 IGD	Shift 1 IGD	Shift 1 BPJS	Shift 2 PU	Shift 2 IGD	Shift 2 IGD	Shift 3 IGD
1	2	2	0	1	9	4	5	7
2	6	1	2	0	8	6	5	7
3	4	0	0	0	5	0	0	8

4.3.2.2 Mutasi

Metode mutasi yang digunakan adalah *reciprocal exchange mutation*. Digunakannya *reciprocal exchange mutation* pada penelitian ini karena metode tersebut hanya memilih 2 posisi (exchange position) secara random yang kemudian kedua posisi tersebut ditukarkan. *Offspring* yang dihasilkan dari proses reproduksi *mutasi* dengan metode yaitu 0.2 x 3 = 0.6 ≈ 1 maka *offspring*-nya sebanyak 1. Tahapan yang dilakukan pada proses mutasi ini yaitu dengan menukar 2 gen secara acak pada *parent* yang telah ditentukan. *Parent* yang digunakan untuk proses mutasi ini adalah P3. Gen yang mengalami pertukaran dapat dilihat di Tabel 4.5.

Tabel 4.5 Proses Mutasi pada P2

P2

Hari	Shift 1 PU	Shift 1 IGD	Shift 1 IGD	Shift 1 BPJS	Shift 2 PU	Shift 2 IGD	Shift 2 IGD	Shift 3 IGD
1	9	4	0	0	3	2	10	8
2	6	2	10	4	11	7	9	8
3	2	0	0	0	7	0	0	8

C3

Hari	Shift 1 PU	Shift 1 IGD	Shift 1 IGD	Shift 1 BPJS	Shift 2 PU	Shift 2 IGD	Shift 2 IGD	Shift 3 IGD
1	9	4	0	0	3	2	8	10
2	6	11	10	4	2	7	9	8
3	7	0	0	0	2	0	0	8

4.3.2.3 Evaluasi

Pada tahap evaluasi, akan dilakukan perhitungan *fitness* untuk semua kromosom yang telah terbentuk. Nilai *fitness* yang ditetapkan pada penelitian ini seperti pada Persamaan 4.1.

$$Fitness = \frac{100}{1+pinalti}, \text{ dimana pinalti} = \sum Bp \sum Np \quad (4.1)$$

Keterangan:

∑Bp adalah jumlah bobot pelanggaran

∑Np adalah jumlah nilai pelanggaran

Perhitungan *fitness* yang dilakukan ini berdasarkan dari jumlah perhitungan batasan (*constraint*) yang telah ditentukan sebelumnnya. *Constraint* merupakan batasan yang tidak boleh dilanggar agar dapat menghasilkan

penyusunan penjadwalan yang baik. Apabila terdapat pelanggaran yang dilakukan pada proses penjadwalan, maka akan diberikan nilai pinalti berdasarkan pelanggaran apa yang terlah dilakukan.

Pelanggaran yang dilakukan terdapat 2 jenis, yaitu *hard constraint* dan *soft constraint*. *Hard constraint* adalah pelanggaran berat atau jenis batasan yang harus dihindari dalam pembuatan jadwal, sedangkan *soft constraint* adalah jenis pelanggaran atau batasan yang dalam pembuatannya masih bisa ditoleransi keberadaannya namun sedapat mungkin dihindari untuk medapatkan solusi penjadwalan yang terbaik. Apabila jumlah pelanggaran yang dilakukan semakin sedikit, maka penjadwalan yang dilakukan semakin baik. Namun apabila pelanggaran yang dilakukan semakin banyak, maka dapat diartikan bahwa optimasi penjadwalan yang dilakukan belum cukup baik. Berikut pada Tabel 4.6 merupakan batasan(*constraint*) pada permasalahan penjadwalan jaga dokter di IGD.

Tabel 4.6 *Constraint* pada Penjadwalan Jaga Dokter di IGD

No.	Pelanggaran	Jenis Pelanggaran	Bobot Pelanggaran	Nilai Pelanggaran
1.	Tidak boleh muncul 2 nama dalam 1 shift	*Hard Constraint*	4	1
2.	Tidak boleh bekerja dengan shift yang berurutan	*Soft Constraint*	1	1
3.	Tidak boleh bekerja lebih dari 1 shift dalam sehari	*Soft Constraint*	1	1

Bardasarkan contoh perhitungan nilai *fitness* yang telah dihitung dengan menyertakan nilai *constraint, maka* pada Tabel 4.7 akan ditunjukkan hasil nilai *fitness* untuk setiap kromosom yang terbentuk.

C2

Hari	Shift 1 PU	Shift 1 IGD	Shift 1 IGD	Shift 1 BPJS	Shift 2 PU	Shift 2 IGD	Shift 2 IGD	Shift 3 IGD
1	2	2	0	1	9	4	5	7
2	6	1	2	0	8	6	5	7
3	4	0	0	0	5	0	0	8

Keterangan :

: *Hard Constraint*

: *Soft Constraint*

Fitness pada C2

$$= \frac{100}{(1+((hard\ constraint*np)+(soft\ constraint*np)+(soft\ constraint*np))}$$

$$= \frac{100}{(1+((4*1)+(1*1)+(1*1))}$$

= 52,63157895

Tabel 4.7 *Fitness* pada Kromosom

Individu	*Fitness*
P1	16,66666667
P2	100
P3	11,11111111
C1	7,142857143
C2	14,28571429
C3	50

Semakin tinggi nilai *fitness* yang didapatkan dalam perhitungan, maka akan semakin baik solusi yang dihasilkan. Sebaliknya apabila semakin kecil nilai *fitness* yang didapatkan dalam perhitungan, maka akan semakin buruk terhadap solusi yang dihasilkan. Nilai pinalti merupakan nilai yang berbanding terbalik dengan nilai *fitness*, semakin kecil nilai pinalti yang didapat maka semakin baik nilai *fitness* yang di dapatkan.

4.3.2.4 Seleksi

Pada penelitian yang sedang dilakukan ini, seleksi yang digunakan adalah seleksi *elitism*. Seleksi *elitism* adalah seleksi yang memilih nilai terbaik dari hasil *fitness* yang telah dilakukan pada proses evaluasi (Tabel 4.7), dan pada Tabel 4.8 merupakan hasil *fitness* terbaik yang akan membentuk populasi yang baru.

Tabel 4.8 Seleksi *Elitism* berdasarkan Nilai *Fitness*.

Individu	Individu Awal	Nilai Fitness
P1	P2	100
P2	C3	50
P3	P1	16,66666667

4.4 Perancangan Pengujian Algoritme

Perancangan pengujian algoritme ini digunakan sebagai sarana penggambaran pengujian algoritme yang akan dilakukan pada bab selanjutnya. Pengujian yang dilakukan pada optimasi penjadwalan dokter di IGD menggunakan algoritme genetika ini adalah pengujian konvergensi jumlah generasi, nilai *popSize*, dan pengujian pada nilai *cr* dan *mr*.

4.4.1 Pengujian Jumlah *PopSize*

Pada pengujian ini, dilakukan pengujian nilai *popSize* untuk mengetahui berapa nilai *popSize* yang paling optimal untuk penyusunan jadwal jaga dokter di IGD menggunakan algoritme genetika. Perancangan pengujian nilai *popSize* yang dilakukan menggunakan nilai *popSize* 10, 20, 30, 40, 50, 60, 70, 80, 90, dan 100. Sedangkan jumlah percobaan yang digunakan untuk perhitungan rata-rata *fitness*

adalah sebanyak 10 kali. Pada Tabel 4.9 akan digambarkan perancangan untuk pengujian nilai *popSize*.

Tabel 4.9 Pengujian *PopSize*

No.	Nilai *Popsize*	Nilai *fitness* percobaan ke-*i*										Rata-rata nilai *fitness*
		1	2	3	4	5	6	7	8	9	10	
1.	10											
2.	20											
3.	30											
4.	40											
5.	50											
6.	60											
7.	70											
8.	80											
9.	90											
10.	100											

4.4.2 Pengujian Nilai Generasi

Pada pengujian ini, dilakukan pengujian jumlah generasi untuk mengetahui berapa banyak generasi yang paling optimal untuk penyusunan jadwal jaga dokter di IGD menggunakan algoritme genetika. Perancangan jumlah generasi yang dilakukan menggunakan banyak generasi 50, 100, 150, 200, 250, 300, 350, 400, 450 dan 500. Sedangkan jumlah percobaan yang digunakan untuk perhitungan rata-rata *fitness* adalah sebanyak 10 kali. Pada Tabel 4.10 akan digambarkan perancangan untuk pengujian jumlah generasi.

Tabel 4.10 Pengujian Jumlah Generasi

No.	Jumlah Generasi	Nilai *fitness* percobaan ke-*i*										Rata-rata nilai *fitness*
		1	2	3	4	5	6	7	8	9	10	
1.	50											
2.	100											
3.	150											
4.	200											
5.	250											
6.	300											
7.	350											
8.	400											

9.	450											
10.	500											

4.4.3 Pengujian Kombinasi Nilai *Crossover Rate (cr)* dan *Mutation Rate (mr)*

Pada pengujian ini, dilakukan pengujian kombinasi nilai *crossover rate (cr)* dan *mutation rate (mr)* untuk mengetahui berapa kombinasi nilai *crossover rate (cr)* dan *mutation rate (mr)* yang paling optimal untuk penyusunan jadwal jaga dokter di IGD menggunakan algoritme genetika. Perancangan pengujian kombinasi nilai *crossover rate (cr)* dan *mutation rate (mr)* menggunakan jumlah percobaan sebanyak 10 kali. Pada Tabel 4.11 akan digambarkan perancangan untuk pengujian kombinasi nilai *crossover rate (cr)* dan *mutation rate (mr).*

Tabel 4.11 Pengujian Kombinasi Nilai *Crossover Rate (cr)* dan *Mutation Rate (mr)*

No.	Kombinasi		Nilai *fitness* percobaan ke-*i*										Rata-rata nilai *fitness*
	cr	*mr*	1	2	3	4	5	6	7	8	9	10	
1.	1	0,1											
2.	0,9	0,2											
3.	0,8	0,3											
4.	0,7	0,4											
5.	0,6	0,5											
6.	0,5	0,6											
7.	0,4	0,7											
8.	0,3	0,8											
9.	0,2	0,9											
10.	0,1	1											

4.5 Perancangan Antarmuka

Perancangan antarmuka merupakan gambaran dari tampilan sebenarnya pada sistem Optimasi Penjadwalan Dokter di IGD Menggunakan Algoritme Genetika. Pembuatan perancangan antarmuka ini juga dapat mempermudah implementasi dari antarmuka sistem yang ada.

4.5.1 Halaman Awal (*home)*

Pada gambar 4.16 akan digambarkan rancangan tampilan antarmuka pada halaman awal. Kemudian akan dijelaskan juga bagian-bagian dari perancangan antarmuka pada halaman awal.

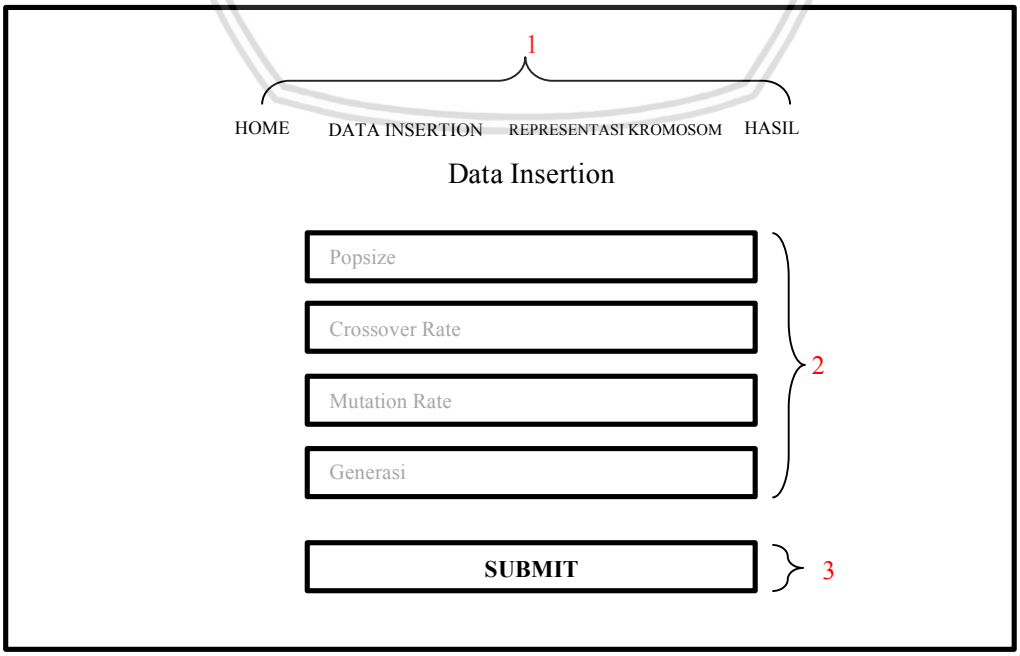
Gambar 4.16 Halaman Awal (*home*)

Penjelasan gambar:

1. Poin 1 adalah menu. Menu tersebut terdiri dari *home, data insertion*, representasi kromosom, dan hasil.
2. Poin 2 adalah kolom untuk judul dari sistem.

4.5.2 Halaman *Data Insertion*

Pada gambar 4.17 akan digambarkan rancangan tampilan antarmuka pada halaman *data insertion*. Kemudian akan dijelaskan juga bagian-bagian dari perancangan antarmuka pada halaman *data insertion*.



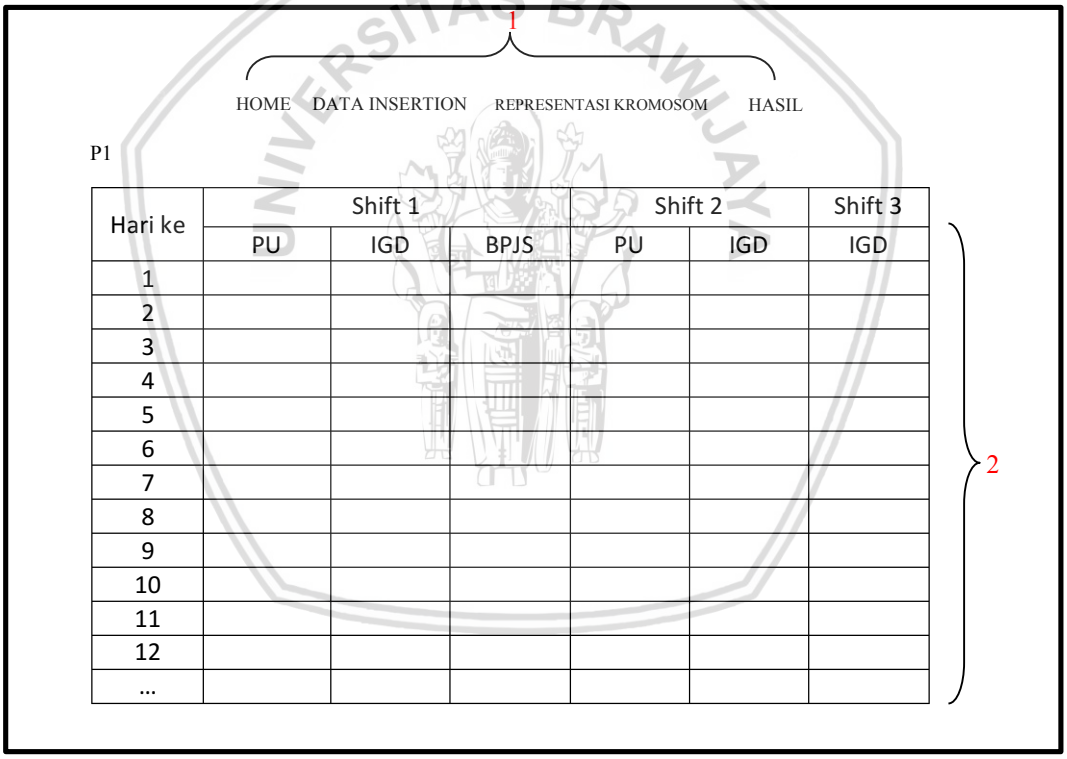
Gambar 4.17 Halaman *Data Insertion*

Penjelasan gambar:

1. Poin 1 adalah menu. Menu tersebut terdiri dari *home*, data *insertion*, representasi kromosom, dan hasil.
2. Poin 2 adalah kolom *text area* yang dapat diisi oleh *user* untuk meng-*input* nilai yang diinginkan.
3. Poin 3 adalah *button* untuk memproses perhitungan dari data yang di-*input* oleh *user* pada kolom *text area* sebelumnya.

4.5.3 Halaman Representasi Kromosom

Pada gambar 4.18 akan digambarkan rancangan tampilan antarmuka pada halaman representasi kromosom. Kemudian akan dijelaskan juga bagian-bagian dari perancangan antarmuka pada halaman representasi kromosom.



Hari ke	Shift 1			Shift 2		Shift 3
	PU	IGD	BPJS	PU	IGD	IGD
1						
2						
3						
4						
5						
6						
7						
8						
9						
10						
11						
12						
...						

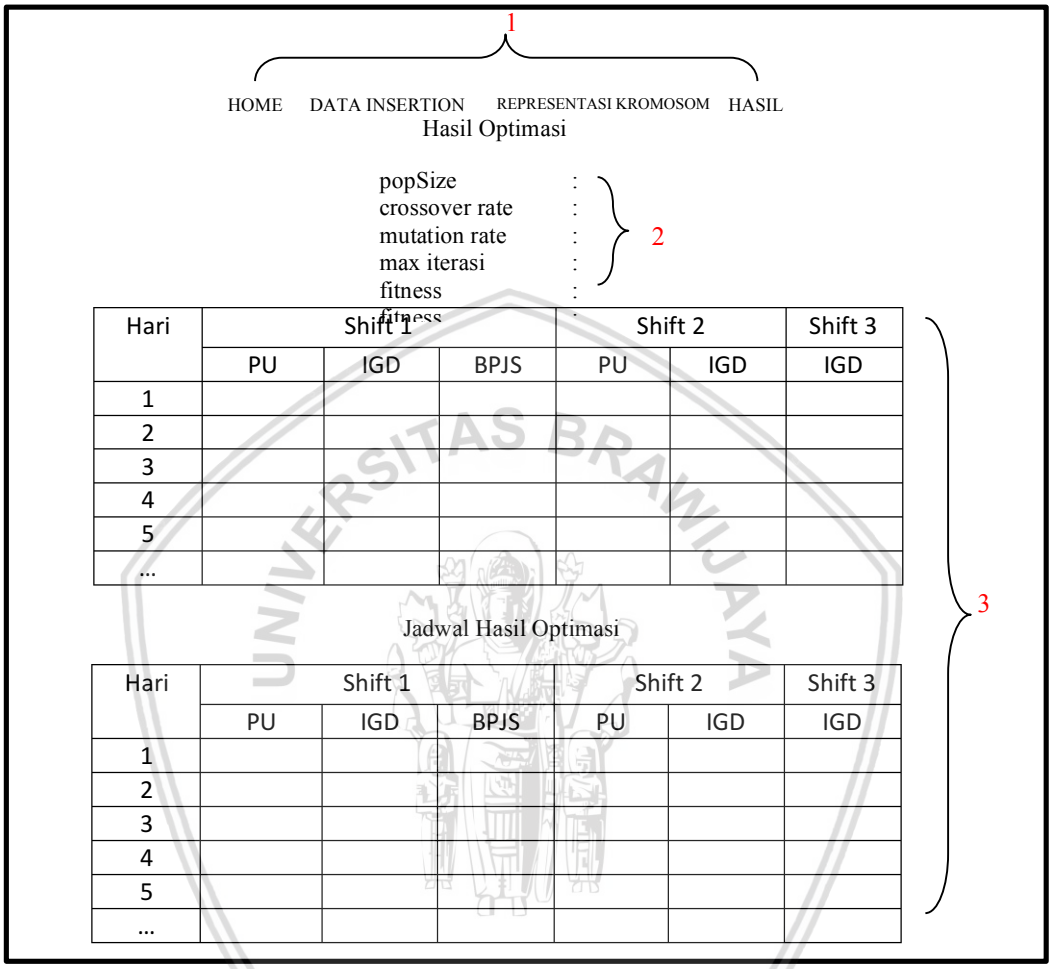
Gambar 4.18 Halaman Representasi Kromosom

Penjelasan gambar:

1. Poin 1 adalah menu. Menu tersebut terdiri dari *home, data insertion*, representasi kromosom, dan hasil.
2. Poin 2 adalah tabel untuk menampilkan representasi kromosom. Jumlah tabel akan bergantung pada berapa jumlah *popsize* yang di-*input* oleh *user.*

4.5.4 Halaman Hasil

Pada gambar 4.19 akan digambarkan rancangan tampilan antarmuka pada halaman hasil. Kemudian akan dijelaskan juga bagian-bagian dari perancangan antarmuka pada halaman hasil.



Gambar 4.19 Halaman Hasil

Penjelasan gambar:

1. Poin 1 adalah menu. Menu tersebut terdiri dari *home, data insertion*, representasi kromosom, dan hasil.
2. Poin 2 adalah tempat untuk menampilkan nilai *popSize, cr, mr, max iterasi* dan fitness yang telah di-*input* sebelumnya pada halaman *data insertion*.
3. Poin 3 adalah adalah tabel untuk menampilkan hasil optimasi dari iterasi yang terbaik.

BAB 5 IMPLEMENTASI

Pada bab ini akan dibahas mengenai implementasi sistem penjadwalan *shift* jaga dokter di IGD berdasarkan proses dari perancangan sistem yang telah dirancang sebelumnya. Pembahasan pada bab ini terdiri dari penjelasan implementasi *source code* Algoritme Genetika dan implementasi antarmuka penjadwalan *shift* jaga dokter di IGD.

5.1 Implementasi Penjadwalan *Shift* Jaga Dokter di IGD

5.1.1 Inisialisasi Populasi Awal

Inisialisasi populasi awal yaitu proses pembentukan individu awal dengan menggunakan representasi kromosom berupa bilangan *integer* yang dibangkitkan secara acak pada pembentukannya. *Source code* tentang inisialisasi populasi awal beserta penjelasannya akan ditunjukkan pada Tabel 5.1.

Tabel 5.1 *Source Code* Inisialisasi populasi awal

No.	Kode Program
1	`function inisialisasi_populasi($hari_awal) {`
	`// $index_shift = ['PU_1', 'IGD_1', 'IGD_12', 'BPJS_1', 'PU_2', 'IGD_2', 'IGD_22', 'IGD_3'];`
2	`$hari = ['senin', 'selasa', 'rabu', 'kamis', 'jumat', 'sabtu', 'minggu'];`
3	`$boleh_kosong = [2, 3, 6];`
	`//$jumlah_hari = 3;`
4	`$panjang_kromosom_perhari = 8;`
5	`$kromosom = [];`
6	`for ($index = 0; $index < $this->popsize; $index++) {`
7	`$counter_hari = array_search($hari_awal, $hari);`
8	`$kromosom_indiv = [];`
9	`$index_kromosom = 0;`
10	`for ($i = 0; $i < $this->jumlah_hari; $i++) {`
11	`$kromosom_perhari = [];`
	`//mencari apakah ada yang kosong`
12	`$jumlah_kosong = rand(0, 3);`
13	`$kosong = [];`
14	`for ($j = 0; $j < $jumlah_kosong; $j++) {`
15	`if (empty($kosong)) {`
16	`$kosong[$j] = rand(0, 2);`
17	`} else {`
18	`$kosong = $this->tambah_unik($kosong, 0, 2);`
19	`}`
20	`}`
	`//membentuk kromosom`
21	`if ($counter_hari == 6) {//jika hari minggu`
22	`for ($j = 0; $j < $panjang_kromosom_perhari; $j++) {`
23	`if ($j == 0 || $j == 4 || $j == 7) {`
24	`$kromosom_perhari[$j + $index_kromosom] = rand(1, 10);`
25	`} else {`
26	`$kromosom_perhari[$j + $index_kromosom] = 0;`
27	`}`
28	`}`
29	`} else {`
30	

```
                    for ($j = 0; $j < $panjang_kromosom_perhari;
                    $j++) {
31                      if ($j == 0 || $j == 4) { //jika pu, boleh
                        iship
32                          $kromosom_perhari[$j + $index_kromosom]
                            = rand(1, 11);
33                      } else {
34                          $kromosom_perhari[$j + $index_kromosom]
                            = rand(1, 10);
35                      }
36                  }
37                  for ($j = 0; $j < count($kosong); $j++) {
38                      $kromosom_perhari[(int) $boleh_kosong[(int)
                        $kosong[$j]] + $index_kromosom] = 0;
39                  }
40              }
41              $index_kromosom += $panjang_kromosom_perhari;
42
43              if ($counter_hari >= 6) {
44                  $counter_hari = 0;
45              } else {
46                  $counter_hari++;
47              }
48              $kromosom_indiv += $kromosom_perhari;
49          }
50          $kromosom[$index] = new individu($kromosom_indiv);
51      }
52      return $kromosom;
53  }
54
55  function tambah_unik($sudah_ada, $min, $max) {
56      $isi = rand($min, $max);
57      while (in_array($isi, $sudah_ada)) {
58          $isi = rand($min, $max);
59      }
60      array_push($sudah_ada, $isi);
61      return $sudah_ada;
62  }
```

Penjelasan *source code* pada Tabel 5.1.

1 : Nama fungsi (inisialisasi_populasi)

2-5 : Inisialisasi variable

6 - 9 : Perulangan *for* untuk mencari hari awal dalam inisialisasi hari

10-13 : Perulangan *for* untuk mencari berapa indeks yang kosong (diacak antara 0-3 indeks)

14-19 : Perulangan *for* untuk mencari indeks ke-berapa yang kosong sesuai dengan perulangan *for* sebelumnya (diacak antara indeks 0-2 atau indeks ke-2, 3, dan 6)

21-28 : Apabila hari minggu, maka *id* dokter hanya diacak dari *id* 1 hingga 10

29-35 : Apabila indeks *array* ke-0 dan ke-4 terisi, maka *id* dokter diacak dari *id* 1 hingga 11.

37-38 : Perulangan *for* untuk mengisi indeks ke-berapa yang bernilai kosong.

43-47 : Kondisi *if else* untuk perhitungan jumlah hari

48 : Proses penjumlahan pecahan-pecahan perhitungan kromosom perhari hingga jumlah hari yang ditentukan di awal

50 : Pembentukan individu baru

55-62 : *Function* tambah_unik digunakan untuk proses pengisian indeks *array* yang bernilai kosong.

5.1.2 Proses *Crossover*

Proses *Crossover* adalah proses reproduksi dengan menyilangkan 2 induk untuk menghasilkan keturunan yang lebih baik. Proses *crossover* yang digunakan adalah *Extended Intermediate Crossover.* Di dalam proses *crossover* ini dibutuhkan nilai *alpha* (*α*) dalam proses perhitungannya. *Source code* tentang crossover beserta penjelasannya akan ditunjukkan pada Tabel 5.2.

Tabel 5.2 *Source Code Crossover*

No.	Kode Program
1	`function crossover($populasi) {`
2	`$anak = [];`
3	`$jumlah_anak = ceil($this->cr * $this->popsize);`
4	`$jumlah_cros = ceil($jumlah_anak / 2);`
5	`$tidak_boleh_kosong=array();`
6	`for ($i = 0; $i < $this->jumlah_hari; $i++) {`
7	`array_push($tidak_boleh_kosong,0+(8*$i));`
8	`array_push($tidak_boleh_kosong,1+(8*$i));`
9	`array_push($tidak_boleh_kosong,4+(8*$i));`
10	`array_push($tidak_boleh_kosong,5+(8*$i));`
11	`array_push($tidak_boleh_kosong,7+(8*$i));`
12	`}`
13	`for ($i = 0; $i < $jumlah_cros; $i++) {`
14	`$index_ind[] = rand(0, count($populasi) - 1);`
15	`$index_ind = $this->tambah_unik($index_ind, 0, count($populasi) - 1);`
16	`$temp_kromosom1 = $populasi[$index_ind[0]];`
17	`$temp_kromosom2 = $populasi[$index_ind[1]];`
18	`$temp_anak = $this->proses_crossover($temp_kromosom1, $temp_kromosom2,$tidak_boleh_kosong);`
19	`array_push($anak, $temp_anak[0]);`
20	`array_push($anak, $temp_anak[1]);`
21	`}`
22	`for ($i = 0; $i < $jumlah_anak; $i++) {`
23	`$anak_cross[$i] = $anak[$i];`
24	`}`
25	`return $anak_cross;`
26	`}`

Penjelasan *source code* pada Tabel 5.2.

1 : Nama fungsi (crossover)

2-5 : Inisialisasi variable

6-11 : Perulangan *for* agar PU dan IGD pada *shift* 1 dan 2 tidak benar-benar kosong

13-21 : Perulangan *for* untuk menentukan individu mana yang akan melakukan *crossover* secara acak

22-24 : Perulangan *for* untuk menjadikan hasil *crossover* menjadi variable $anak

25 : Mengembalikan nilai $anak_cross

Pada *function crossover* terdapat proses yang memanggil *function* proses_crossover. *Source code* tentang proses_crossover beserta penjelasannya akan ditunjukkan pada Tabel 5.3.

Tabel 5.3 *Source Code* proses_crossover

No.	Kode Program
1	`function proses_crossover ($individu1, $individu2, $tidak_boleh_kosong) {`
2	`$kromosom = $individu1->kromosom;`
3	`$kromosom1 = $individu2->kromosom;`
4	`$kromosom_anak1 = [];`
5	`$kromosom_anak2 = [];`
	`//anak pertama`
7	`for ($i = 0; $i < 8; $i++) {`
8	`$alfa[$i] = mt_rand() / mt_getrandmax();`
9	`$alfa1[$i] = mt_rand() / mt_getrandmax();`
10	`}`
11	`$index = 0;`
12	`for ($j = 0; $j < count($kromosom); $j++) {`
13	`$kromosom_anak1[$j] = round($kromosom[$j] + ($alfa[$index] * ($kromosom1[$j] - $kromosom[$j])));`
14	`$kromosom_anak2[$j] = round($kromosom1[$j] + ($alfa1[$index] * ($kromosom[$j] - $kromosom1[$j])));`
15	
16	`if($kromosom_anak1[$j]==0&&in_array($j,$tidak_boleh_kosong)){`
17	`$kromosom_anak1[$j]=$kromosom[$j];`
18	`}`
19	
20	`if($kromosom_anak2[$j]==0&&in_array($j,$tidak_boleh_kosong)){`
21	`$kromosom_anak2[$j]=$kromosom1[$j];`
22	`}`
23	`if ($index == 7) {`
24	`$index = 0;`
25	`} else {`
26	`$index++;`
27	`}`
28	`}`
29	`return [new individu($kromosom_anak1), new individu($kromosom_anak1)];`
30	`}`

Penjelasan *source code* pada Tabel 5.3.

1 : Nama fungsi (proses_crossover)

2-5 : Inisialisasi variable

7-10 : Perulangan *for* untuk mencari nilai acak dari *alpha* (α)

12-14 : Perulangan *for* untuk perhitungan rumus *crossover* dengan menggunakan *extended intermediate crossover*

16-22 : Kondisi *if* untuk pembentukan kromosom anak yang baru dan memastikan agar PU dan IGD pada *shift* 1 dan 2 tidak benar-benar kosong

23-27 : Kondisi *if* untuk menentukan nilai *alpha* hanya sampai indeks ke-7. Apabila jumlah gen lebih dari 7, maka nilainya dikembalikan ke indeks ke-0.

29 : Mengembalikan nilai anak dari hasil *crossover*

5.1.3 Proses Mutasi

Proses mutasi adalah proses reproduksi dengan menyilangkan gen di dalam induk secara random yang dilakukan pada tiap kromosom. Proses mutasi yang dilakukan adalah *reciprocal exchange mutation*. *Source code* tentang mutasi beserta penjelasannya akan ditunjukkan pada Tabel 5.4.

Tabel 5.4 *Source Code* mutasi

No.	Kode Program
1	`function mutasi($populasi) {`
2	`$anak_mutasi = [];`
3	`$jumlah_anak = ceil($this->mr * $this->popsize);`
4	`$index_ind = [];`
5	`for ($i = 0; $i < $jumlah_anak; $i++) {`
6	`if (empty($index_ind)) {`
7	`$index_ind[$i] = rand(0, count($populasi) - 1);`
8	`} else {`
9	`$index_ind = $this->tambah_unik($index_ind, 0, count($populasi) - 1);`
10	`}`
11	`}`
12	`for ($i = 0; $i < count($index_ind); $i++) {`
13	`$anak_mutasi[$i] = $this->proses_mutasi($populasi[$index_ind[$i]]);`
14	`}`
15	`return $anak_mutasi;`
16	`}`

Penjelasan *source code* pada Tabel 5.4.

1 : Nama fungsi (mutasi)
2-4 : Inisialisasi variable
5-11 : Perulangan *for* untuk menentukan gen mana yang akan ditukar secara acak
12-14 : Perulangan *for* untuk mencari lokasi gen dalam 1 populasi
15 : Mengembalikan nilai $anak_mutasi

Pada *function* mutasi terdapat proses yang memanggil *function* proses_mutasi. *Source code* tentang proses_mutasi beserta penjelasannya akan ditunjukkan pada Tabel 5.5.

Tabel 5.5 *Source Code* proses_mutasi

No.	Kode Program
1	`function proses_mutasi($individu) {`
2	`$tidak_boleh_kosong=[0,1,4,5,7];`
3	`$kromosom = $individu->kromosom;`
4	`$anak = [];`
5	`$panjang_kromosom_perhari = 8;`
6	`$index_kromosom = 0;`
7	`for ($i = 0; $i < $this->jumlah_hari; $i++) {`
8	`$temp=[];`
9	`$kromosom_perhari = [];`
10	`for ($j = 0; $j < $panjang_kromosom_perhari; $j++) {`
11	`$kromosom_perhari[$j] = $kromosom[$j + $index_kromosom];`
12	`}`
13	`$index_kromosom += $panjang_kromosom_perhari;`
14	`$index_mutasi = [];`

```
15                      $temp_index_mutasi=[];
16              if ($this->is_minggu($kromosom_perhari)) {
17                  $boleh = [0, 4, 7];
18                  $temp[0] = rand(0, count($boleh) - 1);
19                  $temp = $this->tambah_unik($temp, 0, count($boleh)
                    - 1);
20                  $index_mutasi[0] = $boleh[$temp[0]];
21                  $index_mutasi[1] = $boleh[$temp[1]];
22              } else {
23                  $temp_index_mutasi[0]           =           rand(0,
                    count($tidak_boleh_kosong) - 1);
24                  $temp_index_mutasi              =             $this-
                    >tambah_unik($temp_index_mutasi,                  0,
                    count($tidak_boleh_kosong) - 1);
25                         $index_mutasi[0]                            =
                           $tidak_boleh_kosong[$temp_index_mutasi[0]];
26                         $index_mutasi[1]                            =
                           $tidak_boleh_kosong[$temp_index_mutasi[1]];
27              }
28              $temp = $kromosom_perhari[$index_mutasi[0]];
29              $kromosom_perhari[$index_mutasi[0]]                     =
                $kromosom_perhari[$index_mutasi[1]];
30              $kromosom_perhari[$index_mutasi[1]] = $temp;
31              for ($j = 0; $j < count($kromosom_perhari); $j++) {
32                  array_push($anak, $kromosom_perhari[$j]);
33              }
34          }
35          return new individu($anak);
36  }
```

Penjelasan *source code* pada Tabel 5.5.

1 : Nama fungsi (proses_mutasi)

2-6 : inisialisasi variable

7-12 : Perulangan *for* untuk memecah kromosom keseluruhan jumlah hari menjadi kromosom perhari

16-27 : Kondisi *if else* untuk mencari indeks mana yang akan ditukarkan secara acak

28-30 : Untuk menukarkan indeks yang sebelumnya tlah ditentukan

31-33 : Perulangan *for* untuk menambah nilai mutase yang baru ke variable $anak

35 : Mengembalikan nilai $anak

Pada *function* proses_mutasi terdapat proses yang memanggil *function* is_minggu. *Source code* tentang is_minggu beserta penjelasannya akan ditunjukkan pada Tabel 5.6.

Tabel 5.6 *Source Code* is_minggu

No.	Kode Program
1	`function is_minggu($kromosom) {`
2	`if ($kromosom[1] == 0 && $kromosom[2] == 0 && $kromosom[3] == 0 && $kromosom[5] == 0 && $kromosom[6] == 0) {`
3	`return true;`
4	`} else {`
5	`return false;`
6	`}`
7	`}`

Penjelasan *source code* pada Tabel 5.6.

1 : Nama fungsi (is_minggu)

2-5 : Kondisi *if else* dimana ketika indeks ke 1,2,3,5, dan 6 bernilai 0 maka bernilai benar bahwa itu adalah hari minggu

5.1.4 Proses Evaluasi

Proses evaluasi yang dilakukan merupakan proses perhitungan *fitness* dari setiap populasi, baik populasi yang diinisialisasi di awal maupun populasi hasil proses reproduksi. Perhitungan *fitness* didasarkan ada tidaknya *constraint* di dalam populasi tersebut. *Source code* tentang evaluasi beserta penjelasannya akan ditunjukkan pada Tabel 5.7.

Tabel 5.7 *Source Code* evaluasi

No.	Kode Program
1	`function evaluasi($populasi_awal, $anak_cros, $anak_mutasi) {`
2	`$populasi_akhir = $populasi_awal;`
3	`foreach ($anak_cros as $anak) {`
4	`array_push($populasi_akhir, $anak);`
5	`}`
6	`foreach ($anak_mutasi as $anak) {`
7	`array_push($populasi_akhir, $anak);`
8	`}`
9	`for ($i = 0; $i < count($populasi_akhir); $i++) {`
10	`$fitnes_constrain = $this->hitung_fitnes($populasi_akhir[$i]->get_kromosom());`
11	`$populasi_akhir[$i]->set_fitnes($fitnes_constrain[0]);`
12	`$populasi_akhir[$i]->set_contrain($fitnes_constrain[1]);`
13	`}`
14	`return $populasi_akhir;`
15	`}`

Penjelasan *source code* pada Tabel 5.7.

1 : Nama fungsi (evaluasi)

2 : Inisialisasi variable

3-5 : Perulangan *foreach* untuk menggabungkan antara populasi akhir dengan anak dari proses crossover

6-8 : Perulangan *foreach* untuk menggabungkan antara populasi akhir dengan anak dari proses crossover dan anak dari proses mutasi

9-13 : Perulangan *for* untuk menghitung nilai *fitness* dan untuk *set* nilai *fitness* dan nilai *constrain*

14 : Mengembalikan nilai $populasi_akhir

Pada *function* evaluasi terdapat proses yang memanggil *function* hitung_fitnes. *Source code* tentang hitung_fitnes beserta penjelasannya akan ditunjukkan pada Tabel 5.8.

Tabel 5.8 *Source Code* hitung_fitnes

No.	Kode Program
1	`function hitung_fitnes($kromosom) {`
2	`$bobot1 = 4;`
3	`$bobot2 = 1;`
4	`$bobot3 = 1;`

```
5           $const1 = $this->contrain1($kromosom);
6           $const2 = $this->contrain2($kromosom);
7           $const3 = $this->contrain3($kromosom);
8   return [100 / (1 + ($bobot1 * $const1) + ($bobot2 * $const2) +
    ($bobot3 * $const3)), $const1 + $const2 + $const3];
9   }
```

Penjelasan *source code* pada Tabel 5.8.

1 : Nama fungsi (hitung_fitnes)

2-7 : Inisialisasi variable

8 : Mengembalikan nilai berupa rumus untuk perhitungan *fitness*

Pada *function* hitung_fitnes terdapat proses yang memanggil *function* contrain1. *Source code* tentang contrain1 beserta penjelasannya akan ditunjukkan pada Tabel 5.9.

Tabel 5.9 *Source Code* contrain1

No.	Kode Program
1	`function contrain1($kromosom) {`
2	`        $shift_1 = [0, 1, 2, 3];`
3	`        $shift_2 = [4, 5, 6];`
4	`        $index_kromosom = 0;`
5	`        $panjang_kromosom_perhari = 8;`
6	`        $constrain = 0;`
7	`        for ($i = 0; $i < $this->jumlah_hari; $i++) {`
8	`            $kromosom_perhari = [];`
9	`            for ($j = 0; $j < $panjang_kromosom_perhari; $j++) {`
10	`                $kromosom_perhari[$j] = $kromosom[$j + $index_kromosom];`
11	`            }`
12	`            $index_kromosom += $panjang_kromosom_perhari;`
13	`            if (!$this->is_minggu($kromosom_perhari)) {`
14	`                //shift 1`
15	`                $shift1 = [$kromosom_perhari[$shift_1[0]],`
16	`                $kromosom_perhari[$shift_1[1]],`
17	`                $kromosom_perhari[$shift_1[2]],`
18	`                $kromosom_perhari[$shift_1[3]]];`
19	
20	`                $constrain += $this->cek_shift($shift1);`
21	`                $shift2 = [$kromosom_perhari[$shift_2[0]],`
22	`                    $kromosom_perhari[$shift_2[1]],`
23	`                    $kromosom_perhari[$shift_2[2]]];`
24	`                $constrain += $this->cek_shift($shift2);`
25	`            }`
26	`        }`
27	`        return $constrain;`
28	`}`

Penjelasan *source code* pada Tabel 5.9.

1 : Nama fungsi (contrain1)

2-6 : Inisialisasi variable

7-10 : Perulangan *for* untuk memecah kromosom sejumlah total hari menjadi kromosom perhari

13-25 : Kondisi *if* untuk memecah kromosom perhari menjadi per-*shift*

27 : Mengembalikan nilai $constrain

Pada *function* contrain1 terdapat proses yang memanggil *function* cek_shift. *Source code* tentang cek_shift beserta penjelasannya akan ditunjukkan pada Tabel 5.10.

Tabel 5.10 *Source Code* cek_shift

No.	Kode Program
1	`function cek_shift($data) {`
2	`        $temp1 = [];`
3	`        $total = 0;`
4	`        for ($i = 0; $i < count($data); $i++) {`
5	`            $temp = $data[$i];`
6	`            $temp_array = $data;`
7	`            unset($temp_array[$i]);`
8	`            if (empty($temp1)) {`
9	`                if (in_array($temp, $temp_array)) {`
10	`                    if ($temp != 0) {`
11	`                        $total++;`
12	`                    }`
13	`                }`
14	`                array_push($temp1, $temp);`
15	`            } else {`
16	`                if (in_array($temp, $temp1)) {`
17	`                    continue;`
18	`                } else {`
19	`                    if (in_array($temp, $temp_array)) {`
20	`                        if ($temp != 0) {`
21	`                            $total++;`
22	`                        }`
23	`                    }`
24	`                    array_push($temp1, $temp);`
25	`                }`
26	`            }`
27	`        }`
28	`        return $total;`
29	`}`

Penjelasan *source code* pada Tabel 5.10.

1 : Nama fungsi (cek_shift)

2-3 : Inisiliasisasi variable

4-7 : Perulangan *for* untuk menghapus data yang sama

8-23 : Kondisi *if else* untuk memeriksa setiap *shift* apakah ada data yang sama. Apabila ada yang sama maka *constrain* 1 bernilai 1

28 : Mengembalikan nilai $total

Pada *function* hitung_fitnes terdapat proses yang memanggil *function* contrain2. *Source code* tentang contrain2 beserta penjelasannya akan ditunjukkan pada Tabel 5.11.

Tabel 5.11 *Source Code* contrain2

No.	Kode Program
1	`function contrain2($kromosom) {`
2	`        $jumlah_shift = 3 * $this->jumlah_hari;`
3	`        $shift_sekarang = [$kromosom[0], $kromosom[1], $kromosom[2], $kromosom[3]];`
4	`        $next_shift = 2;`
5	`        $hari_ke = 1;`
6	`        $index = 0;`
7	`        $constrain = 0;`

```
8           for ($i = 1; $i < $jumlah_shift; $i++) {
9               if ($hari_ke == 1) {
10                  $index = 0;
11              } else {
12                  $index = ($hari_ke - 1) * 8;
13              }
14              $shift_next = [];
15              if ($next_shift == 1) {
16                  $shift_next = [$kromosom[0 + $index], $kromosom[1 +
                    $index], $kromosom[2 + $index], $kromosom[3 +
                    $index]];
17                  $next_shift = 2;
18              } else if ($next_shift == 2) {
19                  $shift_next = [$kromosom[4 + $index], $kromosom[5 +
                    $index], $kromosom[6 + $index]];
20                  $next_shift = 3;
21              } else if ($next_shift == 3) {
22                  $shift_next = [$kromosom[7 + $index]];
23                  $next_shift = 1;
24                  $hari_ke++;
25              }
26              $constrain    +=    $this->cek_const2($shift_sekarang,
                $shift_next);
27              $shift_sekarang = $shift_next;
28          }
29          return $constrain;
30  }
```

Penjelasan *source code* pada Tabel 5.11.

1 : Nama fungsi (contrain2)

2-7 : Inisialisasi variable

8-25 : Perulangan *for* untuk memecah kromosom perhari menjadi per-*shift*. Didalamnya terdapat Kondisi *if else* untuk mengetahui indeks ada di *shift* mana

29 : Mengembalikan nilai $constrain

Pada *function* contrain2 terdapat proses yang memanggil *function* cek_const2. *Source code* tentang cek_const2 beserta penjelasannya akan ditunjukkan pada Tabel 5.12.

Tabel 5.12 *Source Code* cek_const2

No.	Kode Program
1	`function cek_const2($shift_sekarang, $shift_next) {`
2	`        $total = 0;`
3	`        for ($i = 0; $i < count($shift_sekarang); $i++) {`
4	`            if ($i == 0) {`
5	`                if (in_array($shift_sekarang[$i], $shift_next)) {`
6	`                    if ($shift_sekarang[$i] != 0) {`
7	`                        $total++;`
8	`                    }`
9	`                }`
10	`            } else {`
11	`                if (in_array($shift_sekarang[$i], $shift_next) && $shift_sekarang[$i] != $shift_sekarang[$i - 1]) {`
12	`                    if ($shift_sekarang[$i] != 0) {`
13	`                        $total++;`
14	`                    }`
15	`                }`
16	`            }`

```
17          }
18          return $total;
19  }
```

Penjelasan *source code* pada Tabel 5.12.

1 : Nama fungsi (cek_const2)

2 : Inisialisasi variable

4-17 : Perulangan *for* untuk untuk memeriksa apakah data tersebut ada pada *shift* selanjutnya atau tidak. Apabila ada, maka nilai total *constrain* akan ditambahkan.

18 : Mengembalikan nilai $total

Pada *function* hitung_fitnes terdapat proses yang memanggil *function* contrain3. *Source code* tentang contrain3 beserta penjelasannya akan ditunjukkan pada Tabel 5.13.

Tabel 5.13 *Source Code* contrain3

No.	Kode Program
1	`function contrain3($kromosom) {//tidak boleh bekerja lebih dari 1 shift dalam sehari`
2	`$index_kromosom = 0;`
3	`$panjang_kromosom_perhari = 8;`
4	`$constrain = 0;`
5	`for ($i = 0; $i < $this->jumlah_hari; $i++) {`
6	`$kromosom_perhari = [];`
7	`for ($j = 0; $j < $panjang_kromosom_perhari; $j++) {`
8	`$kromosom_perhari[$j] = $kromosom[$j + $index_kromosom];`
9	`}`
10	`$index_kromosom += $panjang_kromosom_perhari;`
11	`if (!$this->is_minggu($kromosom_perhari)) {`
12	`//shift 1`
13	`$shift1 = [$kromosom_perhari[0],`
14	`$kromosom_perhari[1],`
15	`$kromosom_perhari[2],`
16	`$kromosom_perhari[3]];`
17	`$shift2 = [$kromosom_perhari[4],`
18	`$kromosom_perhari[5],`
19	`$kromosom_perhari[6]];`
20	`$shift3 = [$kromosom_perhari[7]];`
21	`$constrain += $this->cek_const3(array_unique($shift1), array_unique($shift2), array_unique($shift3));`
22	`} else {`
23	`$shift1 = [$kromosom_perhari[0]];`
24	`$shift2 = [$kromosom_perhari[4]];`
25	`$shift3 = [$kromosom_perhari[7]];`
26	`$constrain += $this->cek_const3($shift1, $shift2, $shift3);`
27	`}`
28	`}`
29	`return $constrain;`
30	`}`

Penjelasan *source code* pada Tabel 5.13.

1 : Nama fungsi (contrain3)

2-4 : Inisialisasi variable

5-9 : Perulangan *for* untuk memecah kromosom sejumlah total hari menjadi kromosom perhari

11-26 : Kondisi *if else* untuk memecah kromosom perhari menjadi kromosom per-*shift*

29 : Mengembalikan nilai $constrain

Pada *function* contrain3 terdapat proses yang memanggil *function* cek_const3. *Source code* tentang cek_const3 beserta penjelasannya akan ditunjukkan pada Tabel 5.14.

Tabel 5.14 *Source Code* cek_const3

No.	Kode Program
1	`function cek_const3($shift1, $shift2, $shift3) {`
2	`$total = 0;`
3	`foreach ($shift1 as $data) {`
4	`if (in_array($data, $shift2) or in_array($data, $shift3)) {`
5	`if ($data != 0) {`
6	`$total++;`
7	`}`
8	`}`
9	`}`
10	`foreach ($shift2 as $data) {`
11	`if (in_array($data, $shift3)) {`
12	`if ($data != 0) {`
13	`$total++;`
14	`}`
15	`}`
16	`}`
17	`return $total;`
18	`}`

Penjelasan *source code* pada Tabel 5.14.

1 : Nama fungsi (cek_const3)

2 : Inisialisasi variable

3-9 : Perulangan *foreach* untuk memeriksa ada tidaknya nama yang sama di *shift* 1 terhadap *shift* 2 dan 3

10-15 : Perulangan *foreach* untuk memeriksa ada tidaknya nama yang sama di *shift* 2 terhadap *shift* 3

17 : Mengembalikan nilai $total

5.1.5 Proses Seleksi

Proses seleksi dilakukan setelah proses perhitungan *fitness* tiap populasi telah selesai. Proses seleksi yang dilakukan adalah seleksi *elitism* yaitu proses pemilihan nilai tertinggi yang akan lolos dan menjadi individu yang terbaik. *Source code* tentang seleksi beserta penjelasannya akan ditunjukkan pada Tabel 5.15.

Tabel 5.15 *Source Code* seleksi

No.	Kode Program
1	`function seleksi($populasi) {`
2	`$populasi_seleksi = [];`
3	`$temp_pop = [];`
4	`for ($i = 0; $i < count($populasi); $i++) {`

```
5               $temp_pop[$i] = $populasi[$i]->get_fitnes();
6           }
7           arsort($temp_pop);
8           $temp_key = [];
9           foreach ($temp_pop as $key => $value) {
10              $temp_key[] = $key;
11          }
12          for ($i = 0; $i < $this->popsize; $i++) {
13              $populasi_seleksi[$i] = $populasi[$temp_key[$i]];
14          }
15          return $populasi_seleksi;
16  }
```

Penjelasan *source code* pada Tabel 5.15.

1 : Nama fungsi (seleksi)

2-3 : Inisialisasi variable

4-6 : Perulangan *for* untuk menampilkan nilai *fitness* dari semua populasi akhir dalam variable $temp_pop

9-11 : Perulangan *foreach* untuk mengambil nilai indeks dari nilai *fitness* yang tertinggi

12-14 : Perulangan *for* untuk mengambil nilai tertinggi sejumlah nilai *popsize* awal yang di-*input*-kan

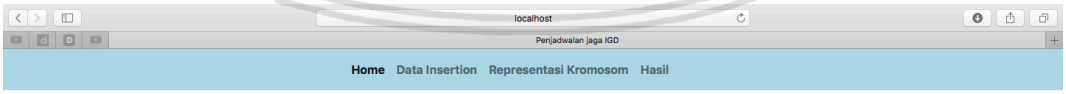
15 : Mengembalikan nilai $populasi_seleksi

5.2 Implementasi Antarmuka

Implementasi antarmuka merupakan hasil implementasi dari perancangan antar muka yang telah digambarkan pada bab sebelumnya. Antarmuka atau *Graphical User Interface (GUI)* dibuat dalam bentuk web menggunakan *css* dengan bahasa pemrograman *PHP*. Antarmuka dibuat untuk mempermudah melihat hasil keluaran program serta membuat tampilan keluaran menjadi semenarik mungkin.

5.2.1 Halaman Awal (*home)*

Implementasi antarmuka dari halaman awal (*home*) dapat dilihat pada Gambar 5.1.

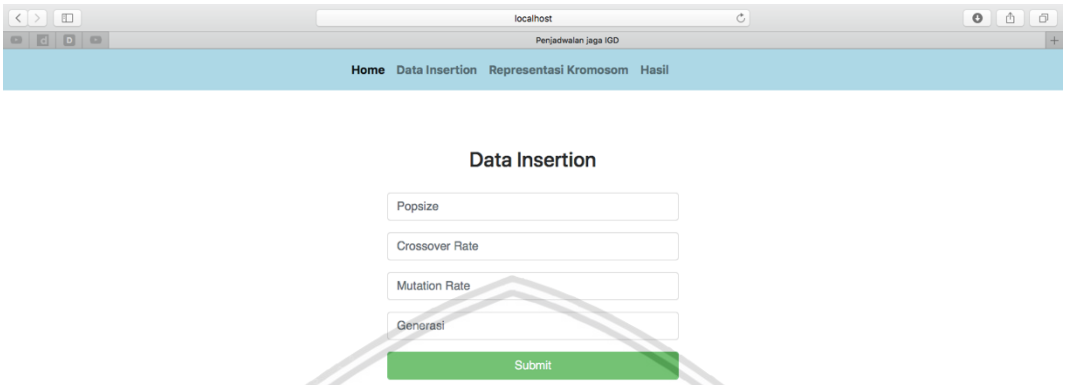


Optimasi
Penjadwalan Shift Jaga
Dokter IGD

Gambar 5.1 Halaman Awal (*home*)

5.2.2 Halaman *Data Insertion*

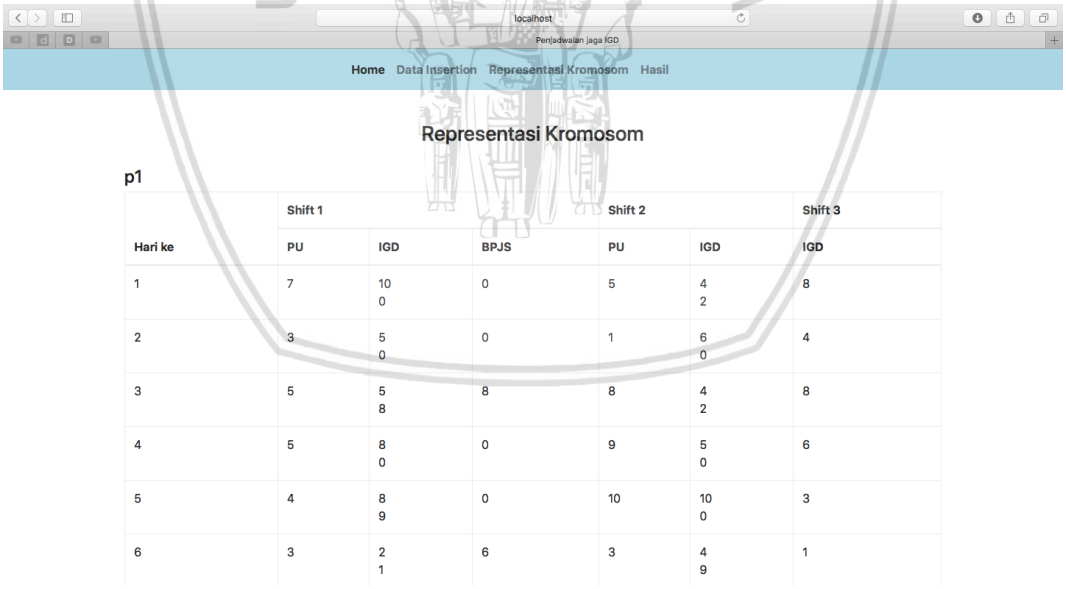
Implementasi antarmuka dari halaman *data insertion* dapat dilihat pada Gambar 5.2.



Gambar 5.2 Halaman *Data Insertion*

5.2.3 Halaman *Data Insertion*

Implementasi antarmuka dari halaman representasi kromosom dapat dilihat pada Gambar 5.3.

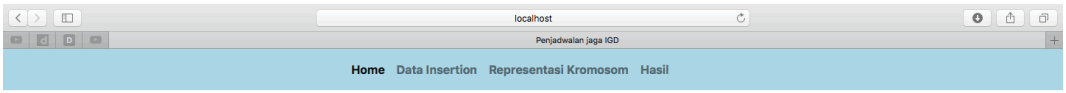


	Shift 1			Shift 2		Shift 3
Hari ke	PU	IGD	BPJS	PU	IGD	IGD
1	7	10 0	0	5	4 2	8
2	3	5 0	0	1	6 0	4
3	5	5 8	8	8	4 2	8
4	5	8 0	0	9	5 0	6
5	4	8 9	0	10	10 0	3
6	3	2 1	6	3	4 9	1

Gambar 5.3 Halaman Representasi Kromosom

5.2.4 Halaman Hasil

Implementasi antarmuka dari halaman hasil dapat dilihat pada Gambar 5.4.



Hasil Optimasi

Popsize = 10

Crossover Rate = 0.6

Mutation Rate = 0.2

Max Iterasi = 2

Fitness : 0.14903129657228

	Shift 1			Shift 2		Shift 3
Hari ke	PU	IGD	BPJS	PU	IGD	IGD
1	10	7 3	1	8	2 1	5
2	7	10 0	0	2	6 0	6
3	8	3 0	0	3	5 10	5
4	3	4 0	0	9	1 0	10
5	3	7 0	0	5	2 0	1
6	1	1	1	6	3	6

Gambar 5.4 Halaman Hasil

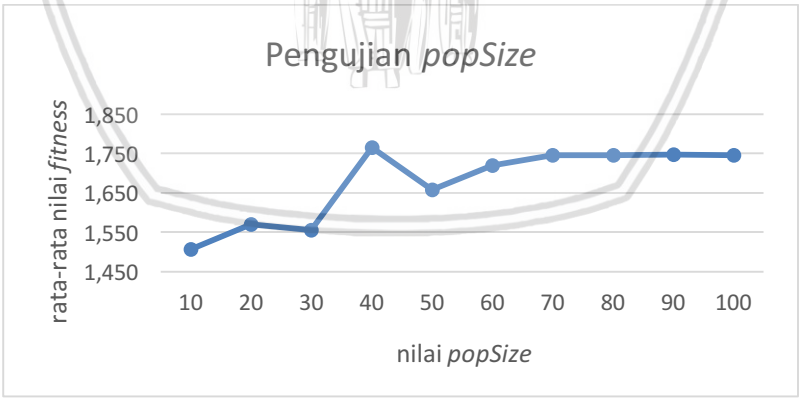
BAB 6 PENGUJIAN DAN ANALISIS

6.1 Pengujian Jumlah *PopSize*

Pengujian jumlah *popsize* ini bertujuan untuk mencari nilai *popsize* terbaik dari banyaknya percobaan yang telah dilakukan. Pengujian ini dilakukan terhadap 10 nilai *popsize* yaitu 10, 20, 30, 40, 50, 60, 70, 80, 90, dan 100 sebanyak masing-masing 10 kali dengan menggunakan banyaknya generasi 50 serta nilai *cr* dan *mr* masing-masing 0.6 dan 0.4. Tabel serta grafik hasil percobaan dapat dilihat pada Tabel 6.1 dan Gambar 6.1.

Tabel 6.1 Hasil Pengujian Nilai *PopSize*

No.	Nilai *Popsize*	Nilai *fitness* percobaan ke-*i*										Rata-rata nilai *fitness*
		1	2	3	4	5	6	7	8	9	10	
1	10	1,613	1,370	1,429	1,449	1,299	1,563	1,389	1,786	1,818	1,351	1,507
2	20	1,563	1,852	1,493	1,538	1,389	1,351	1,639	1,852	1,613	1,408	1,570
3	30	1,695	1,613	1,639	1,538	1,515	1,639	1,471	1,538	1,429	1,471	1,555
4	40	1,887	1,961	1,639	1,754	1,695	1,887	1,695	1,587	1,887	1,667	1,766
5	50	1,587	1,515	1,852	1,613	1,515	1,695	1,408	1,515	2,128	1,754	1,658
6	60	1,818	1,724	1,695	1,667	1,613	1,587	1,961	1,587	1,724	1,818	1,719
7	70	1,695	1,667	1,786	1,695	1,818	1,695	1,961	1,695	1,493	1,961	1,746
8	80	2,041	1,667	1,613	1,563	1,695	1,923	1,754	1,667	1,818	1,724	1,746
9	90	1,587	1,667	1,639	1,695	1,724	2,128	1,852	1,923	1,639	1,613	1,747
10	100	1,667	1,724	1,695	1,587	1,639	1,754	1,724	2,439	1,538	1,695	1,746



Gambar 6.1 Grafik Hasil Pengujian Nilai *PopSize*

Berdasarkan hasil pengujian nilai *popsize* dari sistem penjadwalan *shift* jaga dokter di IGD, didapatkan bahwa rata-rata *fitness* yang cukup rendah terjadi pada percobaan nilai *popsize* 10 dengan nilai 1,507. Kemudian rata-rata *fitness* meningkat tajam pada percobaan nilai *popsize* 40. Setelah itu nilai turun kembali dan pada nilai *popsize* 60. Saat nilai *popsize* mencapai angka 70 hingga 100, rata-rata nilai *fitness* telah mengalami kestabilan atau konvergensi nilai *fitness*.

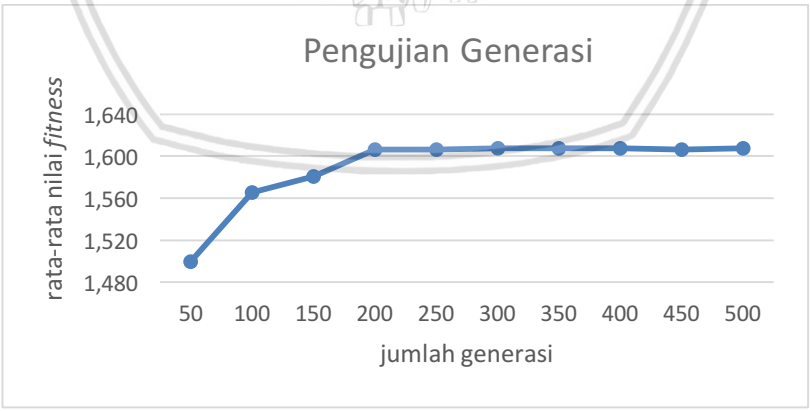
Semakin besar nilai *popsize* maka proses komputasi untuk menampilkan nilai *fitness* akan memakan waktu yang lebih lama dibandikan dengan nilai *popsize* yang lebih kecil.

6.2 Pengujian Nilai Generasi

Pengujian nilai generasi ini bertujuan untuk mencari nilai generasi yang terbaik. Pengujian nilai generasi ini dilakukan terhadap 10 nilai jumlah generasi yaitu 50, 100, 150, 200, 250, 300, 350, 400, 450, dan 500 sejumlah 10 kali percobaan dengan memakai nilai hasil populasi yang terbaik dari pengujian nilai *popsize* sebelumnya yaitu 40 serta nilai *cr* dan *mr* masing-masing 0.6 dan 0.4. Tabel serta grafik hasil percobaan dapat dilihat pada Tabel 6.2 dan Gambar 6.2.

Tabel 6.2 Hasil Pengujian Nilai Generasi

No	Jumlah generasi	Nilai fitness percobaan ke-*i*										Rata-rata nilai fitness
		1	2	3	4	5	6	7	8	9	10	
1	50	1,786	1,587	1,235	1,389	1,515	1,429	1,299	1,282	1,887	1,587	1,500
2	100	1,587	1,754	1,220	1,370	1,408	1,408	1,639	1,449	1,818	2,000	1,565
3	150	1,563	1,887	1,266	1,316	1,563	2,000	1,695	1,299	1,333	1,887	1,581
4	200	1,389	1,250	2,222	1,370	1,205	1,250	2,381	1,471	1,351	2,174	1,606
5	250	1,695	1,471	1,316	1,282	1,389	1,639	1,887	1,333	1,923	2,128	1,606
6	300	1,695	1,587	1,818	1,250	1,351	1,695	1,923	1,235	1,852	1,667	1,607
7	350	1,887	1,639	1,538	1,220	1,563	1,370	1,887	1,333	1,887	1,754	1,608
8	400	2,128	1,667	1,282	1,163	1,266	1,250	1,961	1,389	2,381	1,587	1,607
9	450	2,632	1,176	1,587	1,724	1,493	1,887	1,429	1,408	1,429	1,299	1,606
10	500	1,351	2,083	1,587	1,316	1,235	1,538	2,041	1,429	1,370	2,128	1,608



Gambar 6.2 Grafik Hasil Pengujian Nilai Generasi

Berdasarkan hasil pengujian nilai generasi dari sistem penjadwalan *shift* jaga dokter di IGD, didapatkan bahwa semakin tinggi nilai generasi yang diujikan maka kecenderungan nilai *fitness* untuk meningkat semakin tinggi juga. Ini dapat dibuktikan dengan hasil yang diawali dengan rata-rata *fitness* yang cukup rendah

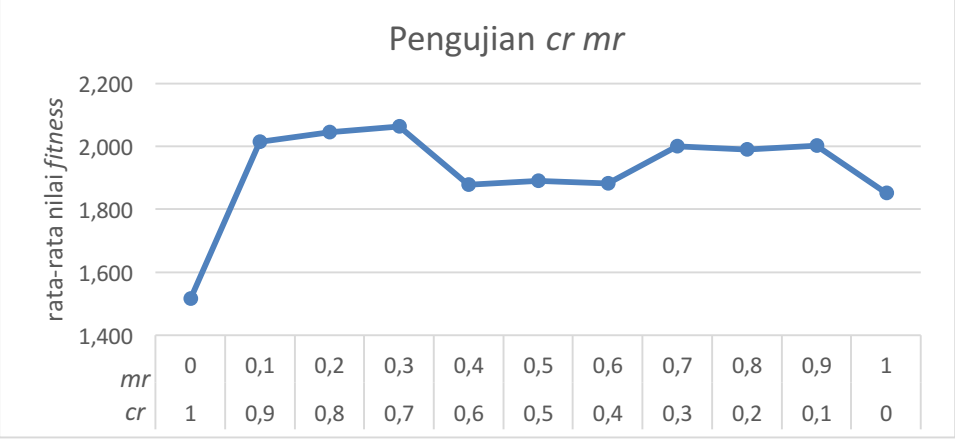
pada percobaan nilai generasi 50. Kemudian rata-rata *fitness* meningkat terus menerus hingga generasi 200. Setelah itu pada nilai generasi 200 hingga 500, rata-rata nilai *fitness* telah mengalami kestabilan atau konvergensi nilai *fitness*. Semakin besar nilai generasi maka proses komputasi untuk menampilkan nilai *fitness* akan memakan waktu yang lebih lama dibandikan dengan nilai generasi yang lebih kecil.

6.3 Pengujian Kombinasi Nilai *Crossover Rate (cr)* dan *Mutation Rate (mr)*

Pengujian kombinasi nilai *crossover rate (cr)* dan *mutation rate (mr)* ini memiliki tujuan untuk mencari nilai *crossover rate (cr)* dan *mutation rate (mr)* yang terbaik. Pengujian nilai *crossover rate (cr)* dan *mutation rate (mr)* ini dilakukan terhadap 11 nilai jumlah *cr* dan *mr* sejumlah 10 kali percobaan dengan memakai nilai hasil dari *popsize* terbaik dari pengujian nilai *popsize* sebelumnya yaitu 40 serta nilai generasi terbaik dari pengujian nilai generasi sebelumnya yaitu 350. Tabel serta grafik hasil percobaan dapat dilihat pada Tabel 6.3 dan Gambar 6.3.

Tabel 6.3 Hasil Pengujian Nilai *Crossover Rate* (*cr*) dan *Mutation Rate* (*mr*)

No.	Kombinasi		Nilai *fitness* percobaan ke-*i*										Rata-rata nilai *fitness*
	cr	*mr*	1	2	3	4	5	6	7	8	9	10	
1	1	0	1,220	1,563	1,449	1,695	1,724	1,316	1,408	1,493	1,923	1,370	1,516
2	0,9	0,1	2,174	2,174	1,818	2,128	2,128	1,493	1,818	1,961	2,222	2,222	2,014
3	0,8	0,2	2,564	2,222	2,083	1,961	1,923	1,818	1,923	2,174	1,786	2,000	2,045
4	0,7	0,3	2,564	1,724	2,128	2,128	2,083	2,174	2,273	1,724	2,174	1,667	2,064
5	0,6	0,4	1,887	1,887	1,563	1,923	2,041	1,724	1,587	2,128	1,923	2,128	1,879
6	0,5	0,5	1,724	1,754	1,818	1,695	1,961	1,818	2,041	2,000	2,273	1,818	1,890
7	0,4	0,6	1,852	1,639	2,128	1,852	1,852	1,818	1,695	2,083	2,083	1,818	1,882
8	0,3	0,7	1,923	1,786	1,818	1,923	2,326	2,273	2,439	2,174	1,786	1,563	2,001
9	0,2	0,8	1,961	2,000	1,923	2,041	2,174	2,041	1,887	1,961	2,000	1,923	1,991
10	0,1	0,9	1,786	1,786	2,083	2,326	1,887	1,961	1,667	2,632	2,174	1,724	2,002
11	0	1	1,887	1,724	1,754	1,818	1,754	2,222	1,695	2,000	1,695	1,961	1,851



Gambar 6.3 Grafik Hasil Pengujian Nilai *Crossover Rate* (*cr*) dan *Mutation Rate* (*mr*)

Berdasarkan hasil dari pengujian nilai *Crossover Rate (cr)* dan *Mutation Rate (mr)* dari sistem penjadwalan *shift* jaga dokter di IGD, terlihat bahwa hasil rata-rata nilai *fitness* dari percobaan yang telah dilakukan terdapat bermacam-macam hasil. Ini terjadi karena nilai dari *cr* dan *mr* sendiri tidak ada ketetapan pasti seperti nilai *popsize* dan nilai generasi. Penentuan nilai *cr* dan *mr* ini sendiri sebenarnya didapatkan agar terjadi keseimbangan kemampuan eksplorasi dan eksploitasi (Lozano & Herrera, 1998). Jika nilai dari cr terlalu rendah dan nilai dari *mr* terlalu tinggi maka Algoritme Genetika tersebut tidak bisa mengeksplorasi ruang pencarian secara efektif. Dan sebaliknya, apabila nilai *cr* terlalu tinggi dan nilai dari *mr* terlalu rendah maka Algoritme Genetika akan memiliki kesempatan eksplorasi ruang pencarian yang semakin sempit (Mahmudy, 2013).

6.4 Analisis Global

Analisis global adalah pengujian akhir yang dilakukan untuk menguji nilai parameter terbaik dari pengujian masing-masing parameter yang telah dilakukan sebelumnya. Pengujian ini juga bertujuan untuk mengetahui seberapa optimal sistem dalam menentukan proses penjadwalan *shift* jaga dokter. Pengujian dilakukan dengan membandingkan hasil *fitness* dari data *real* yang didapatkan dari rumah sakit dengan data yang dihasilkan oleh sistem. Di dalam pengujian ini, data yang diambil dari sistem menggunakan nilai *popSize* sebesar 40, banyaknya generasi adalah 350, dan nilai *cr* serta *mr* adalah 0.7 dan 0.3. Tabel mengenai analisis global dapat dilihat pada Tabel 6.4. Serta tabel berisi data *real* dari rumah sakit dan tabel berisi data yang telah dibuat oleh sistem dapat dilihat selengkapnya pada halaman lampiran.

Tabel 6.4 Analisis Global

	Data *Real* dari Rumah Sakit	Data dari Sistem
Jumlah *Constraint 1* (Terdapat 2 *ID* dokter dalam 1 shift/*hard constraint*)	0	0
Jumlah *Constraint* 2 (Terdapat *ID* dokter yang bekerja dengan *shift* berurutan/*soft constraint*)	6	3
Jumlah *Constraint* 3 (Terdapat *ID* dokter yang bekerja lebih dari 1 *shift* dalam sehari /*soft constraint*)	6	5
Jumlah Nilai *Fitness*	7,692	11,111

Berdasarkan hasil perbandingan pada tabel diatas, dapat disimpulkan bahwa nilai *fitness* pada sistem lebih tinggi 3,419 daripada nilai pada data *real* yang diberikan oleh rumah sakit. Dapat dilihat juga perbandingan jumlah *constraint* antara data *real* dengan sistem yang terdapat selisih 3 *constraint* pada *constraint* 2 dan selisih 1 *constraint* pada *constraint* 3. Maka dari itu, sistem optimasi penjadwalan jaga dokter di IGD ini dapat digunakan sebagai acuan untuk pembuatan jadwal jaga dokter IGD di rumah sakit.

BAB 7 KESIMPULAN

7.1 Kesimpulan

Berdasarkan hasil implementasi dan pengujian dari sistem optimasi penjadwalan *shift* jaga dokter di IGD menggunakan algoritme genetika, maka dapat diambil beberapa kesimpulan yakni sebagai berikut.

1. Sistem optimasi penjadwalan *shift* jaga dokter di IGD dapat diimplementasi dengan menggunakan algoritme genetika dengan menggunakan representasi kromosom berbasis kode *ID* dokter yang diacak, memiliki panjang kromosom sebanyak 8(8 merupakan total jumlah *shift* yang ada). Metode yang digunakan dengan menerapkan algoritme genetika yaitu dengan menggunakan metode reproduksi *crossover* yaitu *extended intermediate crossover*, metode reproduksi mutasi menggunakan *reciprocal exchange mutation*, dan metode untuk proses seleksi menggunakan *elitism selection*. Untuk mendapatkan parameter yang optimal yaitu dengan melakukan 3 pengujian yaitu pengujian untuk ukuran *popsize*, pengujian untuk banyaknya generasi, serta pengujian untuk kombinasi *crossover rate* (*cr)* dan *mutation rate* (*mr)*. Nilai *fitness* tertinggi untuk *popSize* 40 adalah 1.766, nilai *fitness* tertinggi untuk generasi ke-350 adalah 1.608, dan nilai *fitness* tertinggi untuk kombinasi *cr*=0.7 serta *mr*=0.3 adalah 2.064.

2. Hasil terbaik yang dapat diperoleh dari pengujian analisis global yang telah dilakukan dari sistem optimasi penjadwalan *shift* jaga dokter di IGD, memiliki hasil yang menunjukkan bahwa nilai *fitness* dari sistem=11,111 lebih besar dari nilai *fitness* pada data real yang diberikan rumah sakit=7,692. Hasil tersebut menunjukkan bahwa parameter algoritme genetika sangat berpengaruh terhadap solusi yang didapat yaitu dengan memilih individu terbaik yang memiliki nilai *fitness* tertinggi untuk dijadikan sebuah solusi dalam permasalahan penjadwalan *shift* jaga dokter di IGD. Semakin besar nilai *fitness* yang dihasilkan dalam suatu kromosom, maka semakin baik pula solusi yang didapatkan. Sedangkan semakin kecil nilai *fitness* yang dihasilkan dalam suatu kromosom, maka semakin buruk solusi yang didapatkan. Maka dari itu, sistem optimasi penjadwalan jaga dokter di IGD ini dapat digunakan sebagai acuan untuk pembuatan jadwal jaga dokter IGD di rumah sakit.

7.2 Saran

Berdasarkan hasil kesimpulan dari sistem optimasi penjadwalan *shift* jaga dokter di IGD menggunakan algoritme genetika, maka saran yang dapat diberikan yakni sebagai berikut.

1. Sistem optimasi penjadwalan *shift* jaga dokter di IGD yang dibuat menggunakan algoritme genetika dapat dikembangkan lagi dengan menggunakan metode mutasi, *crossover*, dan seleksi yang lain untuk mendapatkan solusi yang lebih bervariasi dan lebih baik.
2. Perlu dikembangkan juga pemilihan parameter yang tepat sehingga akan didapatkan hasil yang lebih optimal dalam waktu yang lebih singkat. Misalnya, untuk menghasilkan nilai kombinasi *cr* dan *mr* yang terbaik dapat digunakan nilai parameter yang adaptif.

DAFTAR PUSTAKA

Atmasari, 2010. Penjadwalan Perawat Unit Gawat Darurat Dengan Menggunakan Goal Programming.

Dhuha & Suseno, 2017. *Penjadwalan Tenaga Kerja Untuk Tiga Shift Kerja Dengan Pengembangan Metode Algoritma Tibrewala, Philippe Dan Browne.* Lhokseumawe-Aceh, s.n., pp. 298-300.

Gen, M. & Cheng, R., 1997. Genetic Algorithm and Engineering Design.

Ilmi, R. R., 2014. Optimasi Penjadwalan Perawat Menggunakan Algoritma Genetika. *DORO: Repository Jurnal Mahasiswa PTIIK Universitas Brawijaya,* Volume 5.

Indonesia, K. K., 2006. *Standar Pendidikan Profesi Dokter.* Jakarta: KONSIL KEDOKTERAN INDONESIA.

Jamil, M., 2015. Studi Fenomenologi: Pengalaman Keluarga Pasien Dalam Berkomunikasi Dengan Perawat Di Prioritas 2 (P2) Instalasi Gawat Darurat. *Jurnal Kesehatan Hesti Wira Sakti,* Volume 3.

Lozano & Herrera, 1998. Tackling real-coded genetic algorithms: operators and tools for behavioural analysis. *Artificial Intelligence Review,* Volume 12, pp. 265-319.

Mahmudy, W. F. & Rahman, M. A., 2011. Optimasi Fungsi Multi-Obyektif Berkendala Menggunakan Algoritma Genetika Adaptif Dengan Pengkodean Real. *Jurnal Ilmiah KURSOR,* pp. 19-26.

Mahmudy, W. F., 2013. *Modul Matakuliah Algoritma Evolusi.* Malang: PTIIK Univ. Brawijaya.

Malliarou, M. G. G. B. F. E. K. &. Z. S., 2014. Family Perceptions of Intensive Care Unit Nurses' Roles: a Greek per- spective. Volume 2, p. 1–3.

Marbun, Y., 2013. Perbandingan Algoritma Genetika dan Particle Swarm Optimization dalam Optimasi Penjadwalan Matakuliah.

Mutakhiroh, I., Saptono, F., Hasanah, N. & Romi, 2007. *Pemanfaatan Metode Heuristik Dalam Pencarian Jalur Terpendek Dengan Algoritma Semut Dan Algoritma Genetika.* Yogyakarta, s.n., pp. B-33 - B-39.

Rezaeiahari, M. & Khasawneh, M., 2017. An optimization model for scheduling patients in destination medical centers. *Operations Research for Health Care,* October, Volume 15, pp. 68-81.

Rifai, U. A., 2011. Pengembangan Aplikasi Penjadwalan Kegiatan dengan Menggunakan Algoritma Genetika (Studi Kasus: Humas Kementrian Agama RI). *Skripsi Fakultas Sains dan Teknologi 7.*

Sufarnap, E. & Sudarto, 2011. Analisis Optimasi Penjadwalan Jaga Dokter Residen Penyakit Dalam Pada Rumah Sakit Pendidikan. *Jurnal SIFO Mikroskil,* Oktober, Volume 12, pp. 97-104.

Suhartono, E., 2015. Optimasi Penjadwalan Mata Kuliah Dengan Algoritma Genetika (Studi Kasus di AMIK JTC Semarang). *INFOKAM,* pp. 132-146.

Supariasa, I. D. N., Bakri , B. & Fajar, I., 2001. *Penilaian Status Gizi.* Jakarta: Penerbit Buku Kedokteran EGC.

Syarif, A., 2014. *Algoritma Genetika Teori dan Aplikasi.* Yogyakarta: Graha Ilmu.

Ulya, A., 2017. Penggunaan Algoritma Genetik Dengan Pemodelan Dua Tingkat Dalam Permasalahan Penjadwalan Perawat Pada Unit Gawat Darurat Rumah Sakit Umum XYZ Surabaya. *SISFO-Jurnal Sistem Informasi.*

Yaqin, M. A. & Lisbiantoro, T., 2012. Optimasi Penjadwalan Perkuliahan Jurusan Teknik Informatika Universitas Islam Negeri Maulana Malik Ibrahim Malang Menggunakan Algoritma Genetika dengan Metode Seleksi Rank. *MATICS,* pp. 191-197.